NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

INVESTIGATING THE UTILITY OF COUPLING
COCOMO WITH A SYSTEM DYNAMICS
SIMULATION OF SOFTWARE DEVELOPMENT

by

Richard W. Smith

September, 1991

Thesis Advisor: Tarek K. Abdel-Hamid

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (*If applicable*) 55	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (*If applicable*)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (*City, State, and ZIP Code*)	10 SOURCE OF FUNDING NUMBERS Program Element No / Project No / Task No / Work Unit Accession Number

11 TITLE (*Include Security Classification*)
INVESTIGATING THE UTILITY OF COUPLING COCOMO WITH A SYSTEM DYNAMICS SIMULATION OF SOFTWARE DEVELOPMENT (UNCLASSIFIED)

12 PERSONAL AUTHOR(S) Smith, Richard W.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (*year, month, day*)	15 PAGE COUNT
Master's Thesis	From To	September 1991	175

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Cost Estimation, Software Project Management, System Dynamics Simulation Model, COCOMO

19 ABSTRACT (*continue on reverse if necessary and identify by block number*)

Cost estimation of software, in this era of budgetary constraints, is vitally important to the success or failure of a software project. Although there are many cost estimation models available, cost overruns and late deliveries still persist.

Coupling the Constructive Cost Model (COCOMO) and the System Dynamics Model of Software Project Management can provide a tool to study project management over the life of a project, to use sensitivity analysis to enhance COCOMO's cost driver set, and to utilize an automated optimization system for software cost estimation in a single or multi-project environment. This new type of model creates a means to study the multi-project environment and determine what the advantages and disadvantages are to sharing resources between different software projects.

Several 'C' programs were developed, that when interfaced and coupled with the system dynamics model, provide a tool to optimize cost estimates in a two project environment. It also creates an environment to perform extensive sensitivity analysis for the enhancement of COCOMO's cost driver set in the single and two project environment.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
X UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Tarek K. Abdel-Hamid	22b TELEPHONE (*Include Area code*) (408) 646-2686	22c OFFICE SYMBOL AS/AH

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted; All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED

Approved for public release; distribution is unlimited.

Investigating the Utility of Coupling COCOMO
with a System Dynamics
Simulation of Software Development

by

Richard W. Smith
Lieutenant, United States Navy
B.S., United States Naval Academy, 1982

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Department of Administrative Sciences

ABSTRACT

Cost estimation of software, in this era of budgetary constraints, is vitally important to the success or failure of a software project. Although there are many cost estimation models available, cost overruns and late deliveries still persist.

Coupling the Constructive Cost Model (COCOMO) and the System Dynamics Model of Software Project Management can provide a tool to study project management over the life of a project, to use sensitivity analysis to enhance COCOMO's cost driver set, and to utilize an automated optimization system for software cost estimation in a single or multi-project environment. This new type of model creates a means to study the multi-project environment and determine what the advantages and disadvantages are to sharing resources between different software projects.

Several 'C' programs were developed, that when interfaced and coupled with the system dynamic model, provide a tool to optimize cost estimates in a two project environment. It also creates an environment to perform extensive sensitivity analysis for the enhancement of COCOMO's cost driver set in the single and two project environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

Cost estimation of software is one of the few areas in the computer industry that has not progressed with the rapid technological advances and growth experienced by the remainder of the computer industry. Although extensive studies have been completed and many cost estimation models developed, there is still no evidence that the software industry can accurately estimate the cost of a software project. Cost overruns, late deliveries, poor reliability, and user dissatisfaction has created an atmosphere of distrust in cost estimation models. In a recent speech, for example, Air Force General Bernard Randolph characterized software as the "Achilles' heel of weapons development" and continued later stating, "On software schedules, we've got a perfect record; We haven't met one yet."(Kitfield, 1989, p. 29) A report from the 101ST Congress summarized, "that in an increasingly constrained budget environment greater controls must be established to alleviate the continual cost overruns and excessive cost growth of these systems."(Congress, 1989, pp. 1-2) In order to contend with these problems, cost estimation studies were directed toward the understanding of the complete software development process, which included many management

issues. Although some progress has been made in the understanding of this development process, the problems still tend to persist. Studies have discovered that this, so called, "Software Monster" is as much a managerial problem as a technical one (Schlender, 1989, p. 100). Meanwhile, the demand for more reliable and more sophisticated software continues to escalate with the increased dependence of computers in everyday life (Abdel-Hamid, 1989, p. 1426).

Coupling an algorithmic model, such as a Constructive Cost Model (COCOMO), and a Systems Dynamics Simulation Model of Software Project Management may enhance the ability to provide better cost estimations. A coupled computer based modeling system can be used to efficiently and effectively study the effects of altering various objective and subjective management controlled variables on the cost and schedule aspects of a software project. The modeling system can also be used to repeat the same project simulation many times while adjusting different parameters in an effort to determine the variables that are most sensitive to the cost and schedule of a single project.

Many organizations today utilize a matrix organizational structure. In this type of organizational structure many resources are shared by several different projects. The coupled modeling system can create a means to simulate two projects in a shared resource environment that automatically refines the cost and schedule estimates. It can also repeat

the simulation process many times while adjusting different parameters not only to achieve optimization in estimation, but also to determine the variables most sensitive to cost and schedule estimates in a two project environment.

B. OBJECTIVES

Several models have been developed to provide cost estimates for software development projects. The COCOMO model has been one of the most widely studied and used models in cost estimation. The System Dynamics Simulation Model of software development has been developed to study the effects of certain management policies on software project development. These management polices are subjective in nature and are difficult to define. The emphasis of this thesis will focus on the development of a new kind of software estimation model that combines an algorithmic model, COCOMO, with a systems dynamic simulation model. It will initially focus on single software development projects, but will also include studies considering cost estimation in a two project environment. Additionally, this thesis will investigate the addition of management variables as a means to enrich the current set of COCOMO cost drivers. The test cases presented will be proofs to ensure proper operation of the interfaces between models. The experiment presented will be that of limiting one specific resource variable between the two

projects and observing the process of automatic refinement of cost estimates over a series of iterations.

C. THE RESEARCH QUESTION

The primary research question of this thesis is to determine the advantages of coupling COCOMO with a dynamic simulation model of software development. The other important question of this thesis is whether the combined model would enhance our ability to do sensitivity analysis and to determine what, if any, approach is best for optimizing cost estimations in a two project environment.

D. SCOPE

The scope of this thesis is to analyze the usefulness of the composite model concept, and to investigate the utility gained from coupling these two models in both the single and two project environments. The scope of this thesis is not, however, to improve cost accuracy.

E. METHODOLOGY

This thesis follows a series of logical steps in the development and testing of a coupled modeling system. The initial phase consists of designing a small "C" test program to determine the best possible interface between the program and the simulation model. Once an interface is successful, the second phase begins. It includes the development of a "C" program which will operate as a front-end system for the

simulation model and provide the COCOMO cost estimations for effort and schedule. After completion of the program and the interface is successful, testing must occur. The testing includes baseline tests of program algorithms to ensure the data being passed between programs is accurate. This phase of testing includes testing the Basic COCOMO Model, the Intermediate COCOMO Model, and nominal productivity calculations.

The last phase includes the programming and testing of two additional "C" programs. The first program is very similar to the program previously developed. It also includes the front-end system and the COCOMO cost estimations for effort and schedule. The difference between the programs is that the program from the previous phase was developed for a single project environment and the latter provides for the same requirements in the two project environment. After successful completion of this program, a back-end program will be developed to evaluate the results, provide adjustments where necessary, and create an automatic iterative loop process which continually refines cost estimations.

The testing for this phase will include a simple test, as above, to prove the accuracy of program algorithms, but will also include an experiment. The experiment will consist of several tests to determine the opportunity gained by creating an iterative loop process for refinement of cost estimations in a two project environment. In addition, it will also test

the ability of this type of modeling system to do extensive sensitivity analysis by limiting certain shared variables and observing the results under several different conditions.

F. ORGANIZATION OF STUDY

This chapter has discussed the general background and themes which direct this study. The remaining chapters are organized as follows. Chapter II discusses the architecture of the system. This includes background discussions on the dynamic simulation model, COCOMO and the design structure of the integration between the two models. Chapter III describes in-depth the operation of the system. Chapter IV discusses the tests and experiments conducted in this thesis. The tests will show the validity of the program algorithms and the experiment will show the potential of using this coupled modeling system. Chapter V discusses the conclusions and recommendations attained from conducting this study.

II. SYSTEM ARCHITECTURE

A. INTRODUCTION

This system is a coupling of a series of "C" programs and a dynamic simulation model to create an interactive, user-friendly support tool for the purpose of studying and refining cost estimation procedures while gaining a better understanding of software development project management. This chapter first discusses the background of the dynamic simulation model and COCOMO. It then discusses, in detail, the system integration between the two models.

B. A DYNAMIC SIMULATION MODEL OF SOFTWARE DEVELOPMENT

The Dynamic Simulation Model is part of an on going study of software development project management dynamics. The model focuses on four basic subsystems which integrate the management process of software development as well as the production-type functions that constitute the software development life cycle (Abdel-Hamid, 1990, p. 21). The four subsystems include: human resource management; software production; controlling; and planning. The Dynamic Simulation Model is unique in that it is able to integrate key management related software development processes such as scheduling, productivity, and staffing to derive implications and gain

knowledge about behavioral aspects of management in the overall software development process. It is also unique in its use of the feedback principles of system dynamics to structure and clarify the complex web of dynamically interacting variables. Figure 2-1 establishes the interactions and relationships between each of the four subsystems (Agan, 1990, p. 7).

The human resource management subsystem comprises the hiring, training, assimilation, and transfer of the human resource. In this subsystem, the work force is divided into two types of employees, newly hired and experienced. Newly added team members tend to be less productive than experienced members. On the other hand, experienced members productivity is reduced due to the training needs involved in assimilating newly added members into the team. Employee turnover also directly impacts project development. The larger the project the greater the turnover rate. This corresponds with the above productivity discussion of newly added members being less productive.(Abdel-Hamid, 1990, p.22)

Figure 2-1 suggests "work force available" has a direct bearing on the allocation of manpower among the different software production activities in the Software Production Subsystem. The primary software production activities are development, quality assurance, rework and testing.

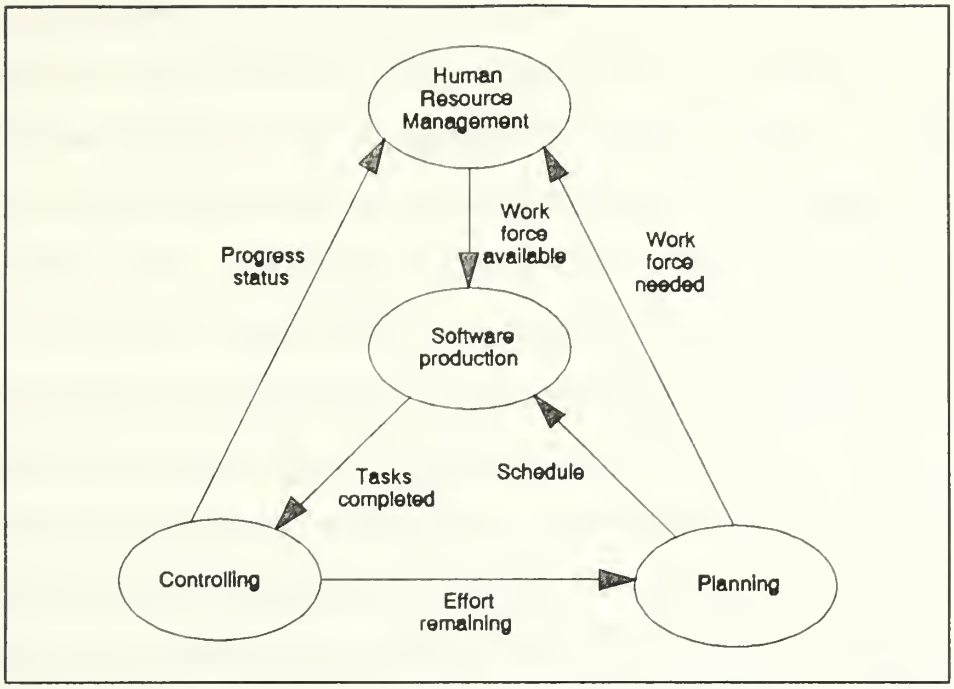

Figure 2-1: Four Sub-systems of the Dynamic Simulation Model

Quality assurance is used as a means of detecting errors in development activities. Although some errors will elude detection until the testing phase, errors detected through quality assurance will be reworked. As progress is made, a comparison of where the project is versus where it should be is evaluated. This evaluation is a function of the control activity in the Controlling Subsystem. Since software is an intangible product during most of the development process and there are no visible milestones to measure progress or quality, the Controlling Subsystem contains some of the most difficult problems a manager must solve. Therefore, the Controlling Subsystem possibly has the greatest impact on this entire system. As depicted in Figure 2-1, the Controlling Subsystem directly effects the Planning Subsystem in the

quantification of "effort remaining" which indirectly impacts both "schedule" and "work force needed" to complete the project. The "progress status" of the project reported back to the human resource management subsystem directly effects team productivity. In early stages of the project, team members rely on the managers assessment of their overall productivity as they are unable to perceive the productiveness of the work force. Therefore, as the project nears completion, the managers projected productivity gradually ceases to influence the perceived productivity of the team as it becomes a function of feedback determined by actual tasks completed.(Abdel-Hamid, 1990, p. 23)

The Planning Subsystem is responsible for the initial project estimates and, when necessary, the revised estimates as each subsystem continues to effect the other until project completion. For example, for a project perceived to be behind schedule, a manager may hire additional employees, delay the schedule, possibly a combination of the two, or do nothing (Abdel-Hamid, 1990, p. 23).

The dynamic simulation environment of this model concentrates on the managerial aspects of software development and on the fundamental understanding of the software development process.

C. COCOMO MODEL

The COnstructive COst MOdel or COCOMO is an algorithmic model that is used to determine initial cost estimates in software development effort and schedule. Initial cost estimates of this model are a function of the estimated size of the software product in source instructions. There are three different versions of COCOMO which include Basic COCOMO, Intermediate COCOMO, and Detailed COCOMO. Basic COCOMO is an algorithmic model that is effective for quick and rough order of magnitude estimates of software costs. However, its accuracy is limited because it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and other project attributes known to have a significant impact on software costs.(Boehm, 1981, p. 58)

The intermediate COCOMO model increases the accuracy of basic COCOMO by incorporating 15 cost drivers into the effort and schedule calculations. These 15 cost drivers are grouped into four categories: software product attributes, computer attributes, personnel attributes, and project attributes. The cost drivers are listed by category below:

- Product Attributes
  - Required Software
  - Reliability
  - Database Size
  - Product Complexity

- Computer Attributes
  - Execution Time Constraint
  - Main Storage Constraint
  - Virtual Machine Volatility
  - Computer Turnaround Time
- Personnel Attributes
  - Analyst Capability
  - Applications Experience
  - Programmer Capability
  - Virtual Machine Experience
  - Programming Language Experience
- Project Attributes
  - Modern Programming Practices
  - Use of Software Tools
  - Required Development Schedule

Each of these cost drivers has an associated multiplying factor used in the algorithmic calculations to determine, with more accuracy, the overall effort and schedule costs of the software development project.(Boehm, 1981, pp. 114-117)

The detailed version of COCOMO employs a three level hierarchical decomposition of the software product whose cost is to be estimated (Boehm, 1981, p. 347). It also uses effort multipliers to determine the phase distribution of effort over the life cycle. This version of COCOMO did not lend itself to this coupled modeling system and, therefore, was not utilized.

In the COCOMO environment there are three modes of software development. They include the Organic Mode, the Semidetached Mode, and the Embedded Mode. Distinguishing between the modes is extremely important to prevent

overestimation or underestimation in the amount of effort required for the project. (Boehm, 1984, p. 20)

The organic mode represents projects with relatively small software teams who operate in a familiar, in-house environment. The teams are comprised of people with extensive experience in the organizations structure, in working with related systems, and in understanding how the system will contribute to the goals and objectives of the organization. The Organic mode environment lends itself to a relatively relaxed atmosphere that leads to higher productivity and smaller diseconomy of scale on the project.(Boehm, 1981, p. 78)

The semidetached mode represents a project that falls between the definition of the organic mode and the embedded mode. The software project development teams are comprised of a wide mixture of experienced and inexperienced people, some of which understand how the system relates to the organization and some that do not.(Boehm, 1981, p. 79)

The embedded mode represents a project which must operate in a strongly coupled complex of hardware, software, regulations and operational procedures. Initially, the team consists of a small group of analysts, normally from outside the organization, who complete product design. Then, again from outside the organization, a large group of programmers are hired to complete the project. Because of rigid requirements and the inability to make changes to these

requirements, the embedded mode environment tends to be less productive and lead to greater diseconomies of scale.(Boehm, 1981, pp. 78-80)

D. INTERFACING "C" PROGRAMS WITH DYNAMIC SIMULATION MODEL

Interfacing between programs is accomplished through the DOS operating system. DOS uses a series of system calls to maneuver through the different processes within the batch files. The DOS operating environment allows the system, through the use of errorlevel calls and goto statements, to execute and exit the different programs which are not compatible due to software language limitations (Schildt, 1988, pp. 140-143).

Figure 2-2 represents a model of the total system architecture which indicates the flows and controls of the execution process of all the programs which makeup the coupled model. The batch file, RUN.BAT is executed by typing RUN at the DOS prompt and pressing enter. This batch file initializes the start up of the coupled model. Once the system is initialized, the user must select the project environment in which to operate. The batch file RUN.BAT first calls the "C" program called MAIN.EXE as shown in Figure 2-2. It is a small program that allows user access into either the single project or two project environment. Figure 2-2 clearly illustrates the distinct paths of the two environments. After the decision is made, the model automatically executes the

selected INPUT program. The INPUT program is used as a front-end system for the dynamic simulation model and as an algorithmic model to complete the COCOMO calculations.

```
**************************RUN.BAT*****************************

/* This is the main batch file which initiates the */
/* execution of the coupled modeling system.       */

     @ECHO OFF
     del report.out
     main
      /* In the "C" language, the exit command with */
      /* an associated number, such as exit(1), enables */
      /* DOS's ERRORLEVEL command to accept the exit */
      /* number, (1) in this case, and call the next */
      /* appropriate program using a GOTO statement.  */
      /* Several examples of this are shown in RUN.BAT, */
      /* the batch file displayed below.  The remainder */
      /* of the "C" programs all interface with the DOS */
      /* batch environment in the same manner.
     IF ERRORLEVEL 1 GOTO two
     input1                    /* if exit (0), then execute */
     IF ERRORLEVEL 1 GOTO done
     GOTO roll
     :roll
     call exec1.bat      /* single project environment */
     output1
     IF ERRORLEVEL 4 GOTO disp
     IF ERRORLEVEL 3 GOTO prn
     IF ERRORLEVEL 0 GOTO done
     GOTO done
     :two
     input2
     IF ERRORLEVEL 1 GOTO done
     GOTO run
     :run
     call exec2.bat      /* two project environment */
     IF ERRORLEVEL 1 GOTO done
     output2
     IF ERRORLEVEL 4 GOTO disp
     IF ERRORLEVEL 3 GOTO prn
     IF ERRORLEVEL 1 GOTO run
     IF ERRORLEVEL 0 GOTO done
     :disp                     /* display on screen */
     type report.out
     GOTO done
     :prn                      /* print results */
     print report.out
     GOTO done
     :done
     REM Program operation is complete.

************************************************************
```

After the inputs and calculation are completed, the program automatically terminates, and the appropriate EXEC batch file is called. The process of interfacing the program to the batch file is discussed in the documentation portion of the RUN.BAT source code displayed below and throughout the source code of the actual programs included in the Appendices.

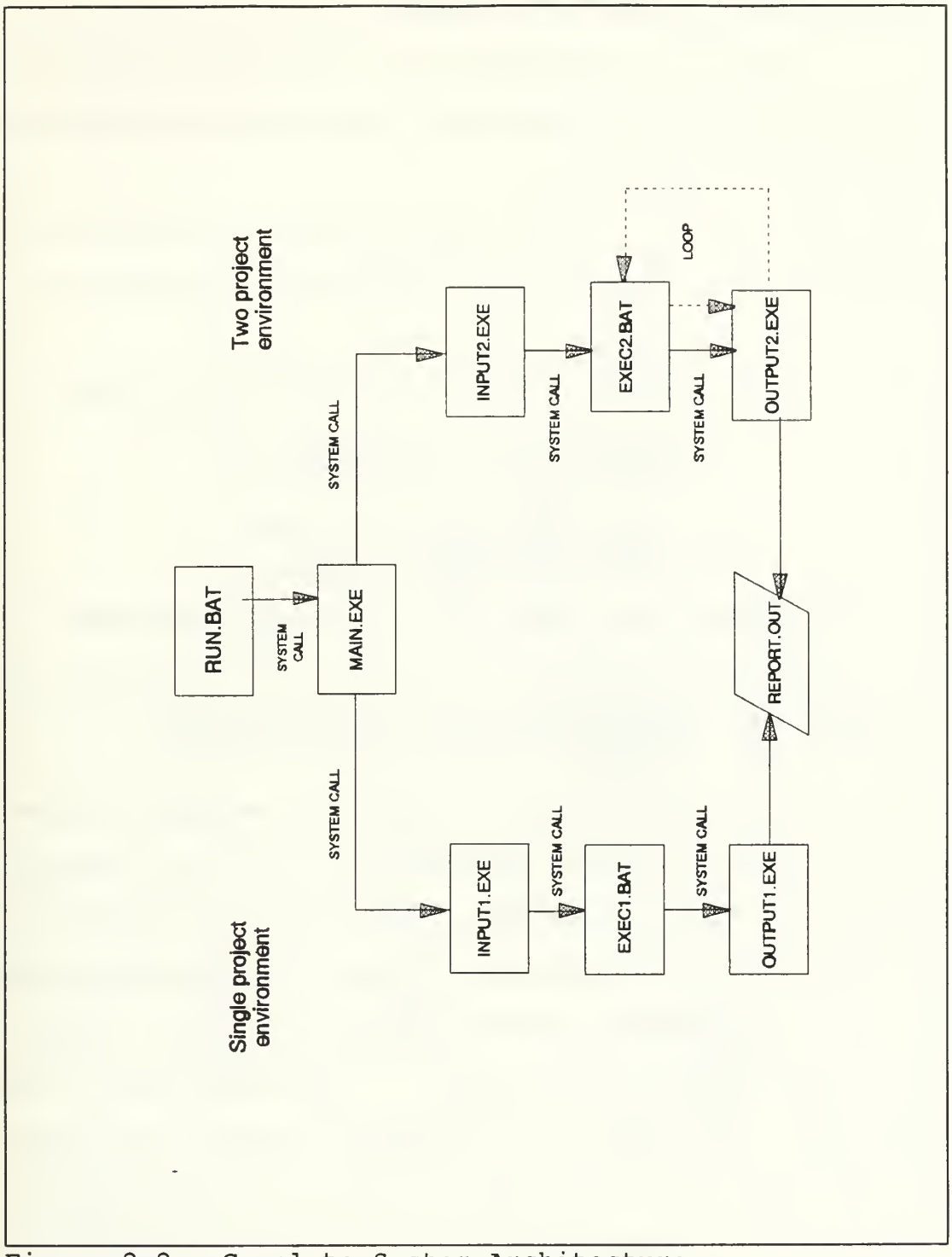

Figure 2-2: Complete System Architecture

1. SINGLE PROJECT ENVIRONMENT

Figure 2-4 represents the processes and data flows within the system architecture that occur in the single project environment.

The single project environment is entered by user selection from a menu which is displayed in Figure 2-3.

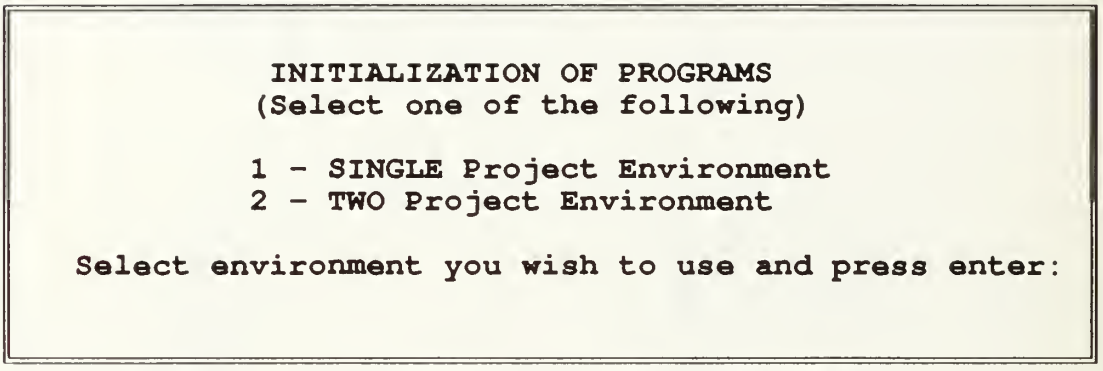

Figure 2-3: INITIALIZATION OF PROGRAMS MENU

Once selected, INPUT1.EXE, a "C" program designed to interface with a single project environment version of the dynamic simulation model, is executed. The INPUT1.EXE program enables the user to utilize current parameters from a saved project, change parameters from a previously saved project or completely enter a new project. INPUT1.EXE has two basic functions. The first is to complete COCOMO cost estimation calculations from the input variables entered by the user. The second is to provide a front end system that allows the user to easily input or provide changes to the input variables

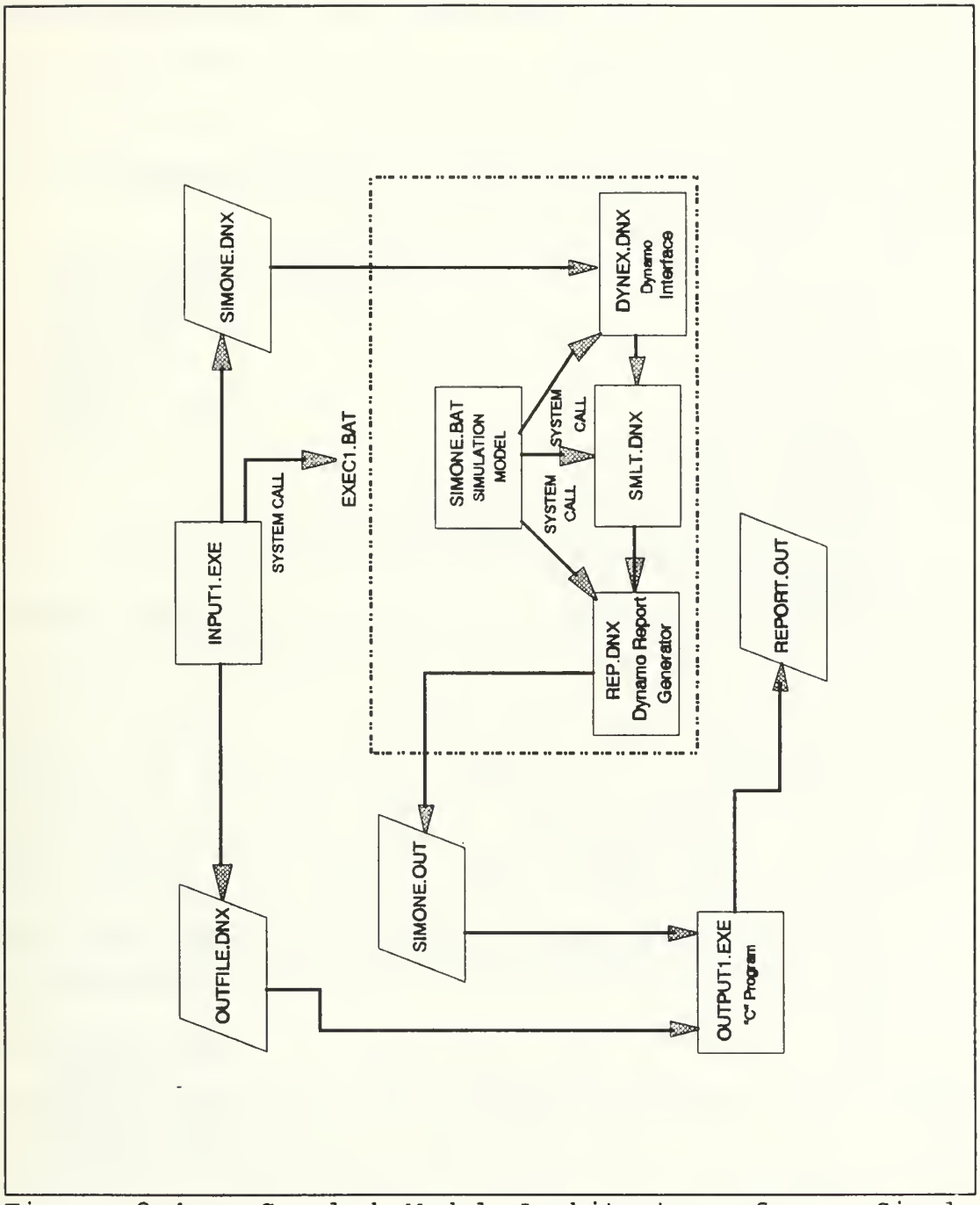

Figure 2-4: Coupled Model Architecture for a Single Project Environment

of the dynamic simulation model. Once the inputs and calculations are completed the entire list of variables are saved to a file called SIMONE.DNX, which is displayed in the input/output file below.

```
************************SIMONE.DNX***************************

/*  This an example of the output file created by   */
/*  INPUT1.EXE and is the input for the simulation. */

     C RJBDSI=64000
     C TOTMD1=4113.23
     C TDEV1=302.60
     C INUDST= 0.50
     C ADMPPS= 1.00
     C HIREDY=40.00
     C AVEMPT=1000000.00
     C TRPNHR= 0.20
     C ASIMDY=80.00
     T TNERPK=25.00 23.86 21.59 15.90 13.60 12.50
     T TPFMQA=0.150 0.150 0.150 0.150 0.150 0.150
              0.150 0.150 0.150 0.150
     C DEVPRT= 0.80
     C DSIPTK=55.22

************************************************************
```

OUTFILE.DNX is a binary file which is also saved at this point in the process. This file is utilized in conjunction with OUTPUT1.EXE. This file passes COCOMO estimated effort and schedule costs to the OUTPUT1.EXE program for report processing. This file was developed to easily pass variables directly from INPUT1.EXE to OUTPUT1.EXE while keeping them completely separate from the dynamic simulation model. After the files are saved, INPUT1.EXE automatically terminates and returns to the DOS environment to call EXEC1.BAT which initiates the dynamic simulation environment

project. This batch file works identical to the main batch file in regard to the interfacing between programs and processes of the dynamic simulation model. The batch file is shown below and is modeled as an insert in Figure 2-4.

```
**************************EXEC1.BAT*****************************

/*  This batch file represents the batch file that calls     */
/*  the simulation model for the single project environment. */

     DYNEX SIMONE -d model.drs
     IF ERRORLEVEL 4 GOTO ERROR
     SMLT SIMONE -GO = -DTM =
     REP SIMONE -T
     GOTO EXIT
     :ERROR
     ECHO *** ERROR 1 ****
     :EXIT

*****************************************************************
```

DYNEX.EXE, SMLT.EXE, and REP.EXE are executable programs which are programmed in the dynamo language and must be present in the default directory for the System Dynamics Model to operate. The SIMONE.DRS file, displayed below, is used by the dynamo report generator to write the required output which will be displayed in SIMONE.OUT.

```
************************SIMONE.DRS***************************

/*  This file works in conjunction with the dynex program  */
/*  and the dynamo report generator to produce the output  */
/*  file SIMONE.OUT.                                       */

     REPORT
     TIME=MAXTIME,
     FORMAT="1<,15>,16<",PICTURE="ZZZZZZV.99"
     "cummd(",CUMMD,")."
     FORMAT="1<,15>,16<",PICTURE="ZZZZZZV.99"
     " time(",TIME,")."

*****************************************************************
```

SIMONE.DNX, as discussed above and as shown in Figure 2-4, is the output file of INPUT1.EXE and contains all of the input variables used by the simulation model. Figure 2-4 also depicts the relationships of the primary components of the dynamic simulation model and how they relate to both the EXEC1.BAT and RUN.BAT batch files.

Upon completion of the dynamic simulation model and EXEC1.BAT, the system again returns to the RUN.BAT batch file where OUTPUT1.EXE is called. This program utilizes information from OUTFILE.DNX and SIMONE.OUT to allow the user the ability to print or display the comparisons between the estimates and actual costs of effort and schedule. The results are saved to a file called REPORT.OUT. This file will remain resident on disk until the program is re-initialized.

2. TWO PROJECT ENVIRONMENT

Figure 2-5 represents the processes and data flows within the system architecture that occur in the two project environment. The two project environment is entered by user selection from a menu which is displayed after execution of MAIN.EXE and is shown in Figure 2-4. Once selected, INPUT2.EXE, a "C" program designed to interface with a two project environment version of the dynamic simulation model, is executed. The INPUT2.EXE program enables the user to utilize current parameters from saved projects, change

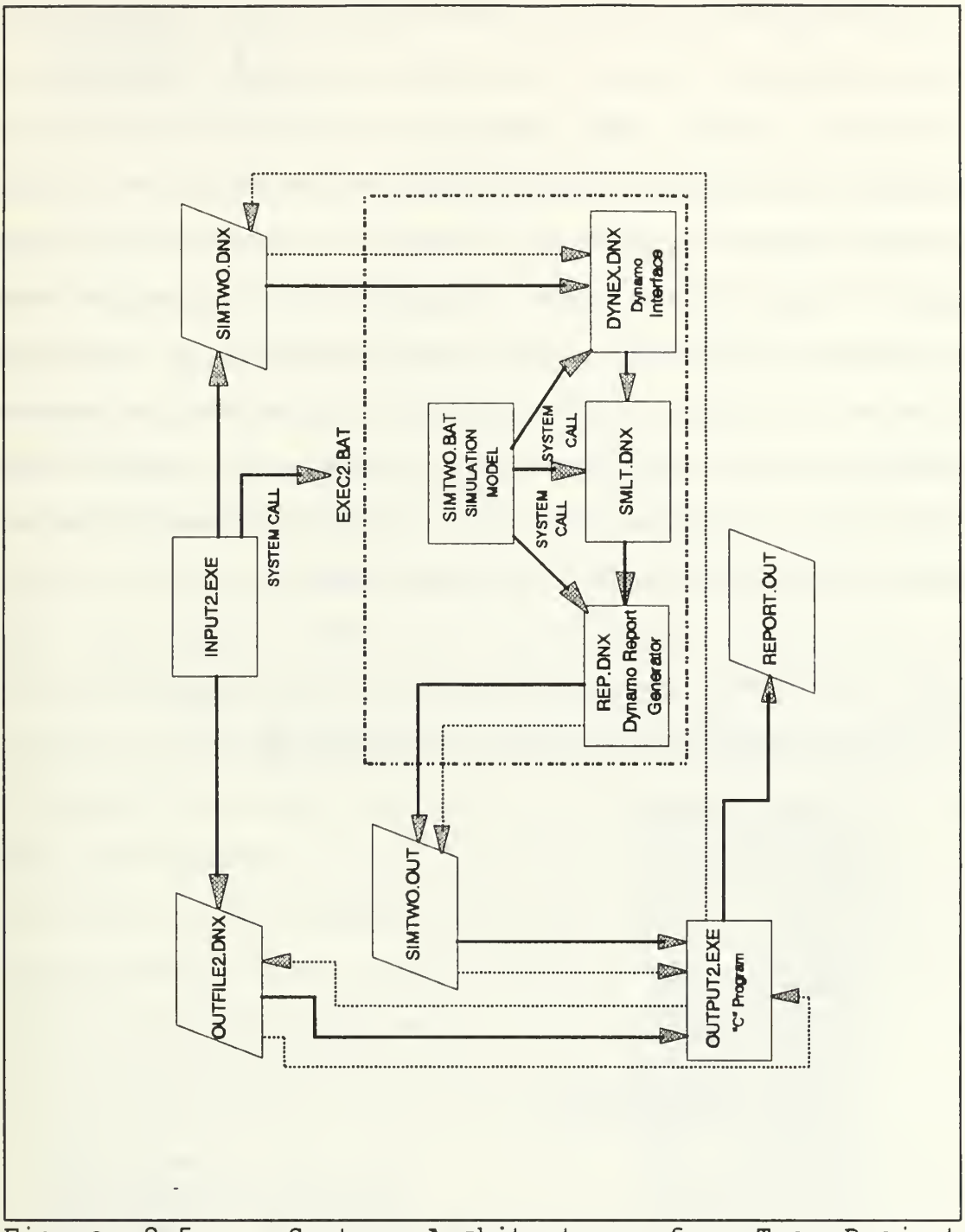

Figure 2-5: System Architecture for Two Project Environment

parameters from previously saved projects or completely enter new projects to complete two basic functions. The first is to complete COCOMO cost estimation calculations from input variables entered by the user for both projects, as discussed, in the previous section. The second is to provide a front end system that allows the user to easily input or provide changes to the input variables for both projects in the dynamic simulation model. Once the COCOMO calculations and inputs are completed and the variables entered, the entire list of variables are saved to a file called SIMTWO.DNX, which is displayed in the input/output file below.

```
**********************SIMTWO.DNX*****************************

/*  This an example of the output file created by   */
/*  INPUT2.EXE and is the input for the simulation. */

     C RJBDSI(1)=64000
     C RJBDSI(2)=64000
     C TOTMD1(1)= 3593
     C TOTMD1(2)= 3593
     C TDEV1(1)=  348
     C TDEV1(2)=  348
     C INUDST(1)= 0.50
     C INUDST(2)= 0.50
     C ADMPPS(1)= 1.00
     C ADMPPS(2)= 1.00
     C HIREDY(1)=40.00
     C HIREDY(2)=40.00
     C AVEMPT(1)=1000000.00
     C AVEMPT(2)=1000000.00
     C TRPNHR(1)= 0.20
     C TRPNHR(2)= 0.20
     C ASIMDY(1)=80.00
     C ASIMDY(2)=80.00
     T TNERP1=25.00 23.86 21.59 15.90 13.60 12.50
     T TNERP2=25.00 23.86 21.59 15.90 13.60 12.50
     T TPFMQ1=0.150 0.150 0.150 0.150 0.150 0.150
              0.150 0.150 0.150 0.150 0
     T TPFMQ2=0.150 0.150 0.150 0.150 0.150 0.150
              0.150 0.150 0.150 0.150 0
     C DEVPRT(1)= 0.80
     C DEVPRT(2)= 0.80
     C DSIPTK(1)=59.89
```

```
     C DSIPTK(2)=59.89
     C STRTDT(1)= 0.00
     C STRTDT(2)= 0.00
     C NCLTWF=1000000.00
**********************************************************
```

OUTFILE2.DNX is a binary file which is also saved at this point in the process. This file is utilized in conjunction with OUTPUT2.EXE. This file passes COCOMO estimated effort and schedule costs to the OUTPUT2.EXE program for report processing and iterative loop control. The file was developed as a means to easily pass variables directly from INPUT2.EXE to OUTPUT2.EXE while keeping them completely separate from the dynamic simulation model process. After the files are saved, INPUT2.EXE automatically terminates and returns to the DOS environment to call EXEC2.BAT. This initiates the dynamic simulation environment for two projects. The batch file works identical to the main batch file and EXEC1.BAT in regard to the interfacing between programs and processes of the dynamic simulation model. EXEC2.BAT is displayed below and is modeled as an insert in Figure 2-5.

```
**************************EXEC2.BAT*****************************

/*  This batch file represents the batch file that calls  */
/*  the simulation model for the two project environment. */


     DYNEX SIMTWO -d model.drs
     IF ERRORLEVEL 4 GOTO ERROR
     SMLT SIMTWO -GO = -DTM =
     REP SIMTWO -T
     GOTO EXIT
     :ERROR
     ECHO *** ERROR 1 ****
     :EXIT

************************************************************
```

DYNEX.EXE, SMLT.EXE, and REP.EXE are executable programs which are programmed in the dynamo language and must be present in the default directory for the System Dynamics Model to operate. The SIMTWO.DRS file, displayed below is used by the dynamo report generator to write the required output which will be displayed in SIMTWO.OUT. SIMTWO.DNX, as discussed above, is the output file of INPUT2.EXE and stores all of the input variables used by the simulation model. Figure 2-5 also depicts the relationships of the primary components of the dynamic simulation model and how they relate to both the EXEC2.BAT and RUN.BAT batch files. Upon

```
**********************SIMTWO.DRS*****************************

/*  This file works in conjunction with the dynex program  */
/*  and the dynamo report generator to produce the output  */
/*  file SIMONE.OUT.                                       */

     REPORT
     TIME=MAXTIME,
     FORMAT="1<,15>,16<",PICTURE="ZZZZZZV.99"
     "cummd(",CUMMD(1),").."
     FORMAT="1<,15>,16<",PICTURE="ZZZZZZV.99"
     "cummd(",CUMMD(2),").."
     FORMAT="1<,15>,16<",PICTURE="ZZZZZZV.99"
     " time(",DURTN(1),").."
     FORMAT="1<,15>,16<",PICTURE="ZZZZZZV.99"
     " time(",DURTN(2),").."

************************************************************
```

completion of the dynamic simulation model and EXEC2.BAT, the system again returns to the DOS environment where OUTPUT2.EXE is called. This program utilizes information from OUTFILE2.DNX and SIMTWO.OUT to create an iterative loop environment.

Figure 2-5 shows the flow of the iterative loop process with dotted lines. The basic theory behind this process is to give the user the ability to decide on the accuracy level and the number of iterations to run in order to continually refine cost estimates. The iterative loop runs as many times as the user desires or until the error rates are equal to or less then those entered by the user. The error rates are determined by using standard algorithms for finding percent error (i.e., difference between estimated effort from COCOMO and actual effort determined by the simulation model divided by the actual effort). In addition to loop limitation and error rate entries, the user may also wish to adjust the actual effort and schedule values determined by the simulation model prior to execution of the next loop. These adjustments can be made to both schedule and effort values in each of the two projects.

The error rates, loop limitations, and adjustment factors give the user the flexibility to run the same projects under many different conditions. This iterative loop process provides the user with an automated tool for sensitivity analysis to gain a better understanding of the software development process and as a means to refine cost estimations. As depicted in Figure 2-5, exiting this process either returns you to the loop or to the report menu for displaying or printing of results from the REPORT.OUT file.

III. SYSTEM OPERATION

A. BACKGROUND

This system is a coupling of a series of "C" programs and a dynamic simulation model to create an interactive, user-friendly support tool for the purpose of studying and refining cost estimation procedures while gaining a better understanding of software development project management.

B. SPECIAL FEATURES

The following is a list of several special features and considerations which were incorporated into the design and development of this system.

- User Friendly: This system was designed for those who have some experience in using COCOMO and the Dynamic Simulation Model. Although the system is designed for ease of use, the user must have a general understanding of the different variables and their associated acronyms in order to achieve effective and efficient data entry.

- Menu Driven: This system uses a menu-driven structure that enables the user to recognize the options available at each level. The highest level menu is the program menu to select a specific environment (i.e., a one or two project environment). Once an environment is selected, the associated function menu appears. This menu enables the user to enter new projects, read existing problems from disk, or exit the program. Several options also have associated sub-menus to help direct the user.

- Easy Data Entry and Modification: This system enables the user to enter data from the keyboard or read stored data from disk. Entry formats are designed for easy entry or modification of input variables.

C. GETTING STARTED

The required and optional hardware, software and peripherals necessary to operate this coupled modeling system include the following:

- Required Equipment: This coupled modeling system is designed to run on an IBM personal computer or true IBM PC compatible computer with at least 256K bytes of memory; at least one high density disk drive; and DOS 3.3 version or higher.

- Optional Equipment: This model supports the IBM enhanced graphics card, IBM color graphics card, IBM VGA card, and Hercules monochrome graphics card. It supports the IBM proprinter and true compatiables. A hard disk drive would improve the overall operation of the system. A math coprocessor is supported for the dynamic simulation model but is not required.

D. GETTING THE SYSTEM READY FOR USE

For ease of use, this system is contained on a single high density floppy diskette. This enables complete operation from almost any floppy drive system. For faster processing of the system, it is recommended to load and operate the system from the hard disk. This system automatically uses the default printer, unless you redirect the output, at the time of printing, to another printer.

1. Installing the Software on Your Hard Disk

If you are familiar with a utility tool, you may wish to create a new directory and copy all files and programs on the diskette to the new directory. Otherwise, continue with the following general procedures.

a. When your screen displays the DOS prompt of the drive you wish to install the program, make a sub-directory using the DOS command 'MKDIR'. For example the sub-directory may be called DSMI for Dynamic Simulation Model Interface.

b. Go to the new sub-directory using the DOS command 'CD'. Then copy all files and programs from the diskette to this new sub-directory using the 'COPY' command.

2. Installation with Hercules Monochrome Graphics Card

a. Ensure the three files "INT10.COM", "HARDCOPY.COM",and "PRINTER.DEF" are copied from the diskette to your hard disk root directory.

b. Add the line "INT10" to your AUTOEXEC.BAT file.

c. Add the line "HARDCOPY" to your AUTOEXEC.BAT file.

E. OPERATING THE SYSTEM

Operating the system can be accomplished either by running the system from the high density floppy diskette or the hard disk. Standard procedures for system operation are as follows:

1. Operating the System from Diskette

The following is a list of steps required to operate the model from a diskette. This system can run on a high density floppy drive system.

a. Turn computer on if not already on.

b. Insert diskette into disk drive and change the default drive to the disk drive which contains the diskette (i.e., if default drive is C then change it so DOS prompt reads, for example, 'A>').

c. At the prompt type 'RUN' and press the ENTER key. The system will be loaded and the initial menu will be displayed. Figure 3-1 shows the initialization menu.

2. Operating the System from the Hard Disk

After the computer is turned on, the interface of the software is designed as a hierarchy of menus and a series of data entry points. To execute the model, proceed with the following steps. At the prompt of the sub-directory you created on the hard drive, type the command 'RUN' and press enter. This will display the welcome screen shown in Figure 3-1. Press any key to continue to the environment selection

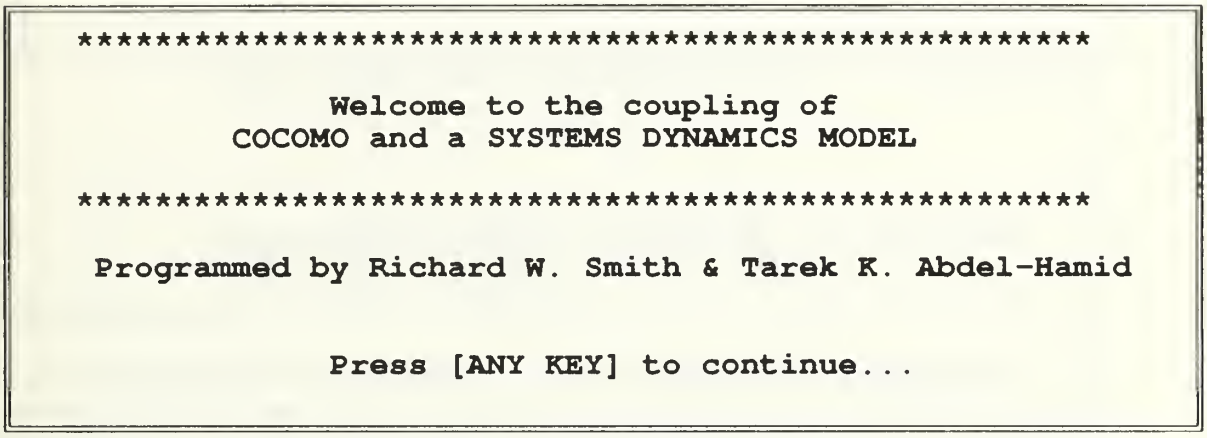

Figure 3-1: Welcome Screen

window similar to the one displayed in Figure 3-2. Your selection will determine which environment you will operate in. Once selected you may not traverse from the single project environment to the two project environment nor vice versa. As mentioned above, this system provides a hierarchy of menus. Thus, when you make a menu selection, you move to a new menu either up or down the hierarchy of menus.

Therefore, whichever environment is chosen, the appropriate initial menu will be displayed similar to the one in Figure 3-3.

```
INITIALIZATION OF PROGRAMS
(Select one of the following)

1 - SINGLE Project Environment
2 - TWO Project Environment

Select environment you wish to use and press enter:
```

Figure 3-2: INITIALIZATION OF PROGRAMS MENU

F. OPERATING IN THE SINGLE PROJECT ENVIRONMENT

The single project environment is available to observe how a single project is affected by software development project management over the life of the project. It can be also used as a tool for sensitivity analysis in studies concerning the effects certain variables have on the development cycle. The baseline data for this model is stored in a data file called BASE.PRF, and provides the values for the SIMONE.DNX input file shown on page 23. Assuming that the COCOMO estimates are accurate, the dynamic simulation model was adjusted, using baseline data, for the purpose of achieving an error rate of less than one percent between the initial cost estimates from COCOMO and the actual results from the simulation model. When

achieved, it can be assumed that the simulation model reflects the management policies of a particular organization (i.e., hiring and firing, turnover rates, training delays, etc.).

Figure 3-3 represents the initial menu in the single project environment. As shown, there are three choices: load a previously saved project, enter a new project, or exit the program. These three options are discussed in detail in the following sections.

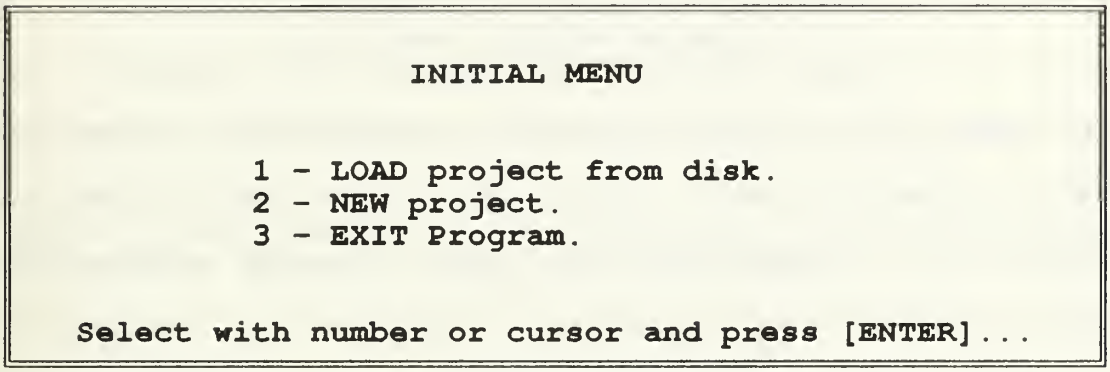

Figure 3-3: INITIAL MENU for Single Project Environment

1. Enter New Project

If you select a new project, a basic instruction page, shown below, is displayed. Selecting any key will allow you to continue.

```
********** IMPORTANT **********

In order to load a NEW project you must enter
input data for both COCOMO and the Dynamic Simulation.
There are two forms on which all data must be entered.
Please enter the data as accurately as possible.

        Press [ANY KEY] to continue...
```

The next step is the data entry phase of the program. A series of variables will be displayed one by one. The program is designed to accept floating point entries for each variable, but it will accept integers and automatically convert them to floats. The system is designed to accept only numbers and a single decimal point for each entry. Other entries could cause the system to malfunction. Recovery from a malfunction is to start over using Ctl-Alt-Del. This shortcoming will be remedied in the next version. If you enter a number incorrectly, continue with the entry process until all the entries are made. You will have the opportunity to re-enter variables in a later step.

There are two entries that are initially made on the COCOMO input page: the size of the project and the name of the file you will store the data. You must enter a file name of your choice with a maximum of eight characters, with the first character being a letter. Do not use a period and extension in your file name. The system automatically adds

the extension ".PRF" to your data file storing the project profile.

The pop-up menu for selecting the basic or intermediate COCOMO Model is displayed in Figure 3-4. Selecting the Basic model initiates the mode selection menu for display and is described below. Selecting the

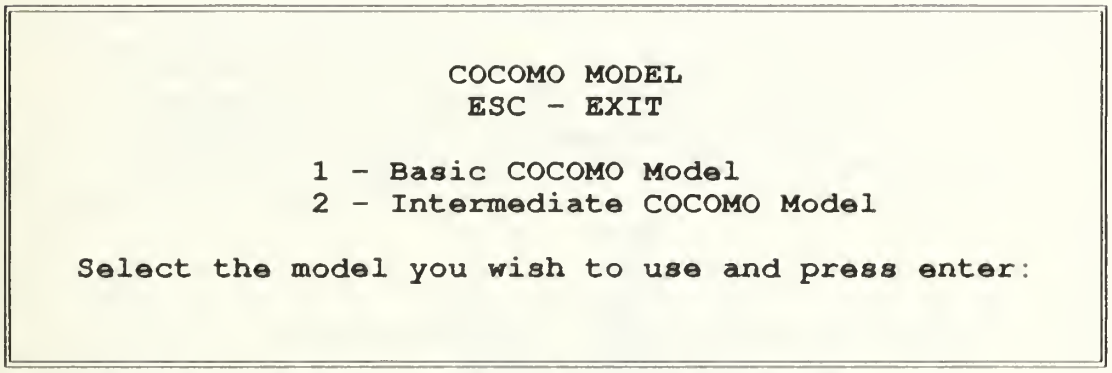

Figure 3-4: Selection Menu for COCOMO Model

intermediate model initiates the screen shown below in Figure 3-5.

This screen is the COCOMO Cost Driver input screen. There are fifteen cost drivers associated with the COCOMO model. They are displayed as acronyms and are all initialized to 1.00. You may leave the cost drivers as they are or change them to the appropriate value. Figure 3-5 shows three different cost drivers that have already been modified. To change the value of a cost driver, you enter the number of the cost driver you wish to change and press enter. A pop-up menu, like the one shown below in Figure 3-6, will be

displayed with all of the levels identified with that particular cost driver (i.e., very high, high, nominal, low etc.) and the associated values available to choose from. Select the value or level you desire and press enter. This returns you to the previous screen, similar to Figure 3-5, with the new value displayed.

```
************************************************
     INTERMEDIATE LEVEL COCOMO MODEL INPUTS
                    for BASE.PRF
************************************************

                 1.  RELY:  0.75
                 2.  DATA:  1.00
                 3.  CPLX:  1.00
                 4.  TIME:  1.00
                 5.  STOR:  1.06
                 6.  VIRT:  1.00
                 7.  TURN:  1.00
                 8.  ACAP:  1.00
                 9.  AEXP:  1.00
                10.  PCAP:  1.00
                11.  VEXP:  1.00
                12.  LEXP:  1.00
                13.  MODP:  1.00
                14.  TOOL:  1.00
                15.  SCED:  1.08

          16.  Press [16 or 0] when entries are complete.

          Select Cost Driver and press [Enter]:
```

Figure 3-5: COCOMO Cost Driver Input Screen

Again you may change as many cost drivers as you wish. Selecting number 16 and pressing return exits you from this screen and displays the mode selection pop-up menu.

The mode selection menu, shown in Figure 3-7, enables you to select the organic mode, the semi-detached mode, or the embedded mode. Whichever mode is selected, the appropriate COCOMO and nominal productivity calculations are completed, and the input display screen is displayed. This screen, shown in Figure 3-8, enables you to display and edit the simulation

```
**********************************************
    INTERMEDIATE LEVEL COCOMO MODEL INPUTS
                for BASE.DAT
**********************************************

        RELY(Required software reliability)
                ESC - EXIT

                1 - Very Low; 0.75
                2 - Low; 0.88
                3 - Nominal; 1.00
                4 - High; 1.15
                5 - Very High; 1.40

          13.  MODP:  1.00
          14.  TOOL:  1.00
          15.  SCED:  1.00

          16.  Press [16 or 0] when entries are complete.

Select Cost Driver and press [Enter]: 1
```

Figure 3-6: Example of Cost Driver Menu

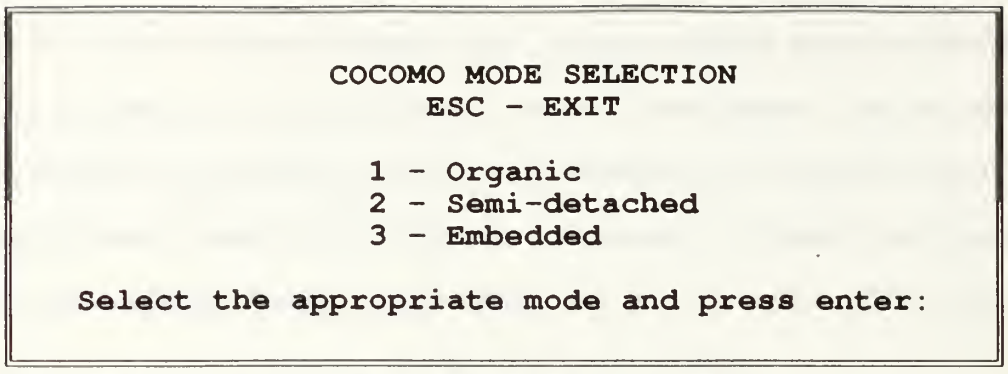

Figure 3-7: COCOMO MODE Selection Menu

model inputs as well as the project size inputs required by COCOMO. By selecting the number of the variable you desire to change and pressing enter, you may now edit any previously entered variable. A single line display will appear on the screen using the full name of the variable instead of the acronym displayed on the previous screen. Enter the new float or integer value and press enter. This returns you back to the full screen shown above. There is no limit on the number of changes or updates you wish to make. When all of the variable values are correct, type 12 and press enter to continue.

After exiting the display/edit screen, the program returns to the Select New Project Menu, as shown in Figure 3-9. You may select Display/Edit to review or edit the new project as described above, Run Dynamic Simulation, or Quit menu and return to the Initial Menu. The Run Dynamic Simulation selection initiates the execution of the dynamic

```
****************************************
        MODEL INPUTS for BASE.PRF
              Organic Mode
****************************************

1. INUDST:  0.500                  8.  (1) TPFMQA[1]:  0.150
2. ADMPPS:  1.000                      (2) TPFMQA[2]:  0.150
3. HIREDY:  40.000                     (3) TPFMQA[3]:  0.150
4. AVEMPT:  1000000.000                (4) TPFMQA[4]:  0.150
5. TRPNHR:  0.200                      (5) TPFMQA[5]:  0.150
6. ASIMDY:  80.000                     (6) TPFMQA[6]:  0.150
7. (1) TNERPK[1]:  25.000              (7) TPFMQA[7]:  0.150
   (2) TNERPK[2]:  23.860              (8) TPFMQA[8]:  0.150
   (3) TNERPK[3]:  21.590              (9) TPFMQA[9]:  0.150
   (4) TNERPK[4]:  15.900             (10) TPFMQA[10]: 0.150
   (5) TNERPK[5]:  13.600          9.  DEVPRT:  0.800
   (6) TNERPK[6]:  12.500         10.  DSIPTK:  59.894

          11. Size of project (KDSI):  64

          12. EXIT and SAVE changes.

Enter number of parameter you wish to change:
```

Figure 3-8: COCOMO Input Display Screen

simulation model. The type equipment you are utilizing determines how long the simulation will take to run. The simulation can take from approximately one minute, with a PC equipped with a co-processor, to approximately 15 minutes, with a PC not equipped with a co-processor. Once running, the only way to exit from the simulation model is to press CTRL-Break.

After the simulation model has run to completion, an Output Selection menu is generated. You may select to display

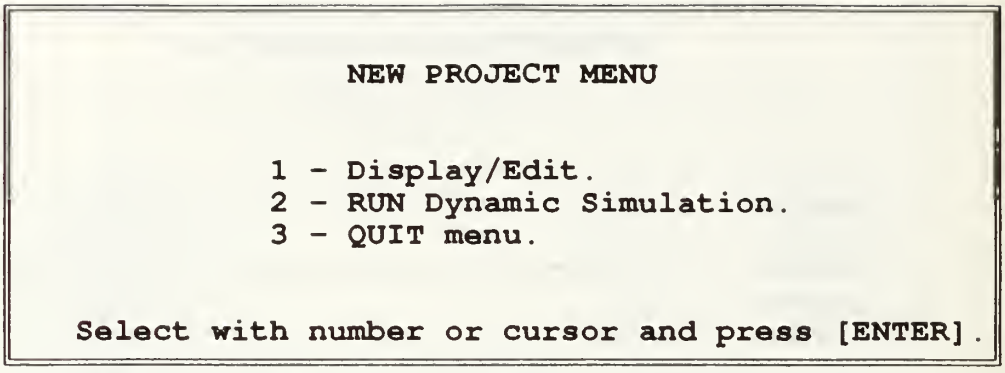

Figure 3-9: NEW PROJECT MENU

the results on screen, print the results, or exit the program as shown in Figure 3-10. The results will remain resident in the REPORT.OUT file until the program is run again. Therefore, if you select exit you still have access to the results. Selecting Display scrolls the results on to the screen. Selecting Print displays a request for the printer you wish to utilize. Either enter the printer or press return to print the results on your default printer.

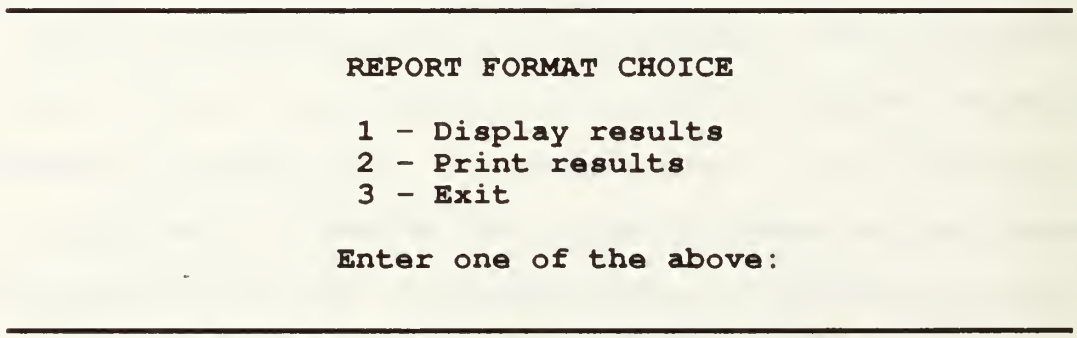

Figure 3-10: REPORT FORMAT SCREEN

2. Load Project from Disk

If you select Load project from disk, the main menu will be displayed, as shown in Figure 3-11. The Main menu gives you four selections to choose from. These include, List projects on disk, Select desired project, Run Dynamic Simulation, or Quit menu. Selecting List will display all of the project data profiles that have been previously saved in the current directory. At the bottom of the list, where requested, enter the file name of the project data profile you

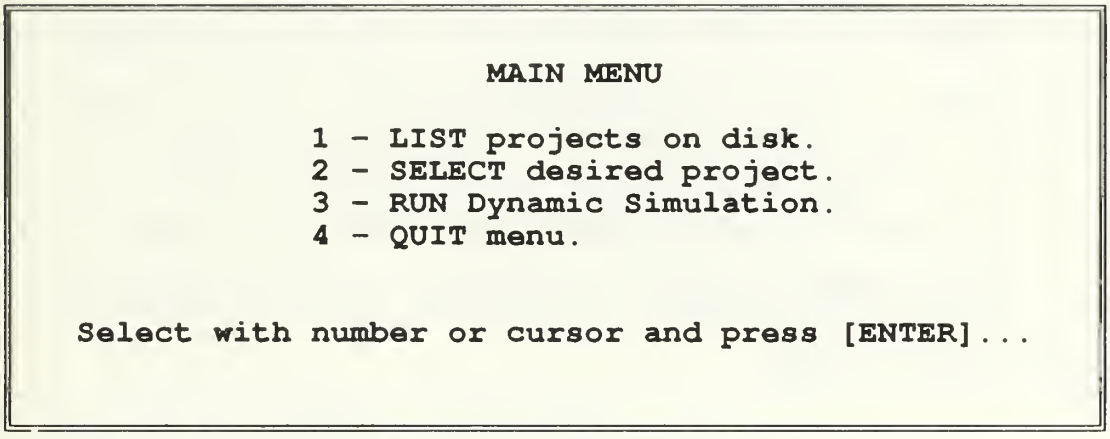

Figure 3-11: MAIN MENU for Single Project Environment

wish to utilize. An example of this screen is displayed below in Figure 3-12.

The display edit screen will automatically be displayed next. The screen is shown in Figure 3-8. You may edit the data as described above or continue using the resident data. If the data profile was saved in the

intermediate COCOMO model, the Cost Driver Input screen will be automatically displayed. This screen, previously shown in Figure 3-5, may also be updated as previously discussed.

Data file listing:

BASE.PRF
HALF.PRF

:> Enter project filename:

Figure 3-12: Current List of Files Screen

After either one or both of these screens have been displayed, the Select Current data profile for COCOMO menu is displayed. This menu, shown in Figure 3-13, allows you to select either the Basic or Intermediate model for COCOMO calculations. This flexibility allows the user to switch between COCOMO models no matter what profile had been previously saved.

Current data file is for the Basic COCOMO
(Select one of the following)

1 - CONTINUE Basic COCOMO Model
2 - Intermediate COCOMO Model

Select the model you wish to use and press enter

Figure 3-13: Select COCOMO Model Menu

You may also change the mode, as discussed above, using the COCOMO Mode Selection menu, previously shown in Figure 3-6, which is displayed next in this part of the menu hierarchy.

The next menu, Figure 3-14, is the Saving Files menu. You can save changes to a datafile under a new name, which ensures the original data remains unchanged, or you may save changes under the same name for updates to the original datafile. After completion of this step, you return to the main menu where you may restart the same process again or quit the menu.

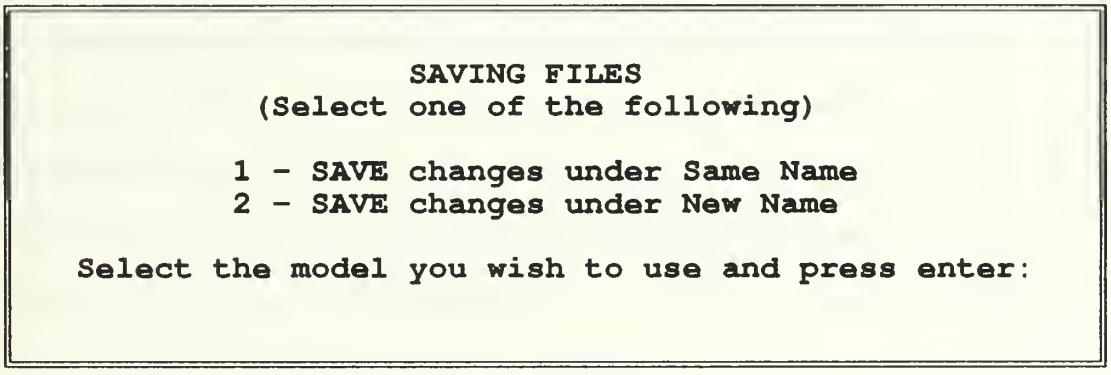

Figure 3-14: SAVING FILES Menu

Selecting SELECT Desired Project displays a single line request for a datafile input. This eliminates the need to display the list described above.

The Run Dynamic Simulation selection initiates the dynamic simulation model for the single project environment.

There is no user interface during the simulation, but as shown in Figure 3-10 above, the simulation concludes with a screen requesting the type of output you would like. Procedures for output are also discussed above.

G. OPERATING SYSTEM IN TWO PROJECT ENVIRONMENT

The two project environment menu hierarchy is almost identical to the single project environment. The basic difference between the two environment menu structures is that you must enter data for two projects instead of one. The repetitiveness of the menu hierarchy is to provide consistency and ease of use for the user. For example, the Main Menu for the Two Project Environment, shown in Figure 3-15, gives you four possible selections to choose from. They include, Select Project 1 from disk, Select Project 2 from disk, Run Dynamic Simulation Model, or Quit menu. For both project selections,

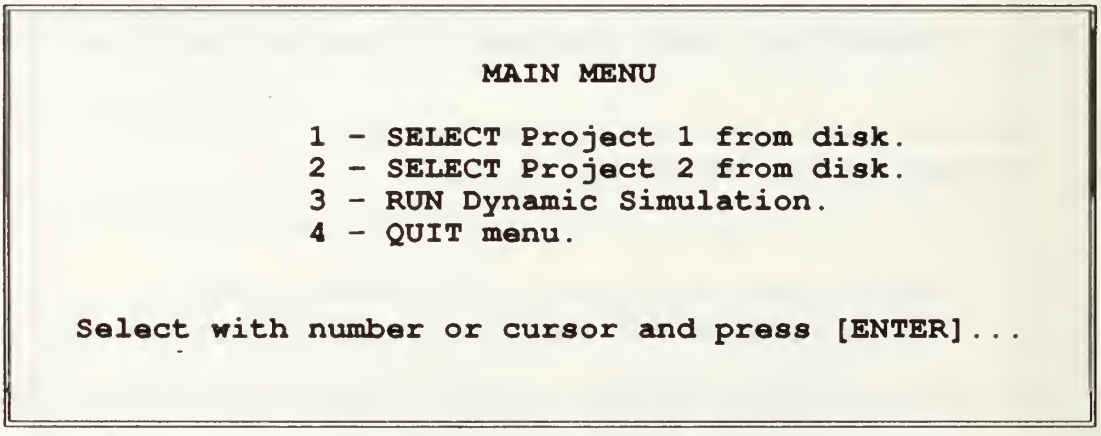

Figure 3-15: MAIN MENU for Two Project Environment

data file profiles are listed, and then followed by a request for the file you wish to use. The program will not allow you

to select Run simulation Model until you have entered both projects as shown below. This is also true if you were

You have not selected Project 1 or Project 2.
Both projects must be selected to run simulation.

Press any key to continue.....

entering new projects. However, the menu structure does allow you to select a project from disk and then go back and enter the other project as new.

The main difference between the two environments is demonstrated when the Run Simulation is selected. In the single project environment, the simulation model is immediately initiated. In the two project environment, there is a series of questions that must be answered prior to the initiation of the simulation model. These questions are shown in Figure 3-16.

This part of the model allows the user to make adjustments to the actual model process. In the two project environment, an iterative loop is built into the model structure which automatically updates the actual results of the simulation model into new estimates. The system then reruns the simulation process with the new estimates. Without knowing how this automatic process will affect certain variables in the simulation, especially regarding productivity, adjustment factors have been included to allow the user the ability to input increased percentages as safety factors or input reduced

The following entries are percentages used to prevent the model from building too much slack into the effort variable in the project. In essence these factors simulate the managers responsibility not to let the productivity lag.

Enter the Effort adjustment factor in project 1 as a percent: 1

Enter the Effort adjustment factor in project 2 as a percent: 1

Enter the Schedule adjustment factor in project 1 as a percent: 1

Enter the Schedule adjustment factor in project 2 as a percent: 1

The following entries allow you to choose the accuracy level and limit the number of loops the model will run before completion.

Enter the accuracy level for Effort as a percent: .01

Enter the accuracy level for Schedule as a percent: .01

Enter the limit of the maximum number of loops the model will do: 2

Figure 3-16: Control Variable Questions

percentages to prevent the system from reducing productivity in areas where there should be little change. If you wish to run the simulation in a nominal mode, or without adjustment, enter a 1.0 for each of the four adjustment factors. Entering a 1.0 is equivalent to running the model at a 100%, entering .95 is equivalent to running the model at 95%, and entering 1.1 is equivalent to running the model at 110%. The above percentages refer to the percent of the effort or schedule estimates which will be entered into the simulation during the iterative loop process. In the nominal mode, the output or actual results of the simulation is used to update the

productivity and as new estimates to be re-entered into the simulation model for the next iteration. The other three entries refer to the accuracy of the results.

Two entries request the error rates of the effort and schedule. This is a calculation of the percent difference between the estimates and the actual results. For example, if you enter a 0.05 for the effort error rate input, the iterative loop will continue re-running the model until the estimated effort going into the simulation model and the actual effort result from the model has an error rate of less than or equal to 0.05. Error rate is the difference between the estimated cost and the actual cost divided by the actual cost. The last entry allows the user to limit the number of loops the model will run. This is necessary if the error rates entered by you are not achieved by the model. The limitation for the number of iterations will prevent an endless loop environment.

H. RESULTS AND REPORTS

The results of this model display the estimated and actual values of effort, schedule, and nominal productivity. The effort and schedule relationships are represented by error rate calculations. In the single project environment there is one line of data in the output, as shown below.

**********REPORT.OUT**********

Estimated man-days	Actual man-days	Percent Error	Estimated Schedule	Actual Schedule	Percent Error
4113.13	4108.86	0.00	301.60	305.00	0.01

**** This data is available in REPORT.OUT ****
**** Each time the model is run REPORT.OUT will change ****

****

Single Project Environment

In the two project environment, after each iterative loop cycle, the output consists of two lines of data, one for each project. A single iteration output is shown below. The results of each run are stored in to a file called REPORT.OUT. The results in this file remain memory resident only until the model is re-initiated using the RUN command from the DOS prompt.

**********REPORT.OUT**********

TOTMD1	CUMMD1	PERCENT ERROR	TDEV1	TIME1	PERCENT ERROR	PRODUCTIVITY OLD	NEW
3593	3591	0.00	348	348	0.00	59.89	59.92

TOTMD2	CUMMD2	PERCENT ERROR	TDEV2	T1ME2	PERCENT ERROR	PRODUCTIVITY OLD	NEW
3593	3591	0.00	348	348	0.00	59.89	59.92

**** This data is available in REPORT.OUT *****
**** Each time the model is run REPORT.OUT will change ****

****

Two Project Environment

IV. TEST AND EVALUATION OF SYSTEM

A. PROJECT DEFINITION

This chapter will provide a description of the algorithms used in determining the COCOMO estimations and the nominal productivity. It will also compare the results of baseline data used by the model with the results of the COCOMO calculations using long hand methodology. This combined model was developed as a tool to aid in the learning process of software development project management. The coupling of these different models and programs has created an environment for an effective and efficient way to study a project over time with the ability to adjust certain variables and conduct sensitivity analysis in determining the variables which are most sensitive to the overall project. The remainder of this chapter will look at two test cases and an experiment to ensure that the calculations in the front-end programs are accurate, to illustrate an example of sensitivity analysis, and to examine an example of the two project environment iterative loop process. All of the following tests and examples were based on data provided in the book by Abdel-Hamid and Madnick (1991).

1. **Test 1**

The following test was run with baseline data to prove the accuracy of the front-end portion of this model in the single project environment. The data used for this test is displayed in the input display screen, shown in Figure 4-1.

```
****************************************
          MODEL INPUTS for BASE.PRF
                Organic Mode
****************************************

1. INUDST:  0.500                  8.  (1) TPFMQA[1]:  0.150
2. ADMPPS:  1.000                      (2) TPFMQA[2]:  0.150
3. HIREDY:  40.000                     (3) TPFMQA[3]:  0.150
4. AVEMPT:  1000000.000                (4) TPFMQA[4]:  0.150
5. TRPNHR:  0.200                      (5) TPFMQA[5]:  0.150
6. ASIMDY:  80.000                     (6) TPFMQA[6]:  0.150
7. (1) TNERPK[1]:  25.000              (7) TPFMQA[7]:  0.150
   (2) TNERPK[2]:  23.860              (8) TPFMQA[8]:  0.150
   (3) TNERPK[3]:  21.590              (9) TPFMQA[9]:  0.150
   (4) TNERPK[4]:  15.900             (10) TPFMQA[10]: 0.150
   (5) TNERPK[5]:  13.600      9.  DEVPRT:  0.800
   (6) TNERPK[6]:  12.500     10.  DSIPTK:  59.894

          11. Size of project (KDSI):  64

          12. EXIT and SAVE changes.

   Enter number of parameter you wish to change:
```

Figure 4-1: Input Display Screen

Most of the variables are direct inputs to the simulation and calculations are not required. The COCOMO calculations vary in nature between the basic and intermediate models. There will be one example of each in test one.

a. Basic COCOMO

Basic COCOMO uses a simple algorithmic methodology to determine the effort and schedule estimations for project development. There are several inputs necessary to accomplish this. The first is the size entry, which is inputted as thousand decision source instruction, KDSI. The other necessary input is a choice between the different modes of COCOMO: organic, semi-detached, and embedded. Each mode has its own series of equations for estimation calculations. Although the organic mode was used in this test, all of the equations for the different modes are displayed in Table 4-1 below.

The first equation uses KDSI to calculate the estimated effort variable. The second equation then uses

TABLE 4-1: BASIC COCOMO EQUATIONS

Basic COCOMO Equations

Mode	Effort	Schedule
Organic	$MM = 2.4(KDSI)^{1.05}$	$TDEV = 2.5(MM)^{0.38}$
Semidetached	$MM = 3.0(KDSI)^{1.12}$	$TDEV = 2.5(MM)^{0.35}$
Embedded	$MM = 3.6(KDSI)^{1.20}$	$TDEV = 2.5(MM)^{0.32}$

the effort variable to determine estimated schedule. The equations from Table 4-1 above represent results calculated in man-months for effort and months for schedule. In the simulation model, the results are calculated in man-days for effort and days for schedule. The conversion to man-days is completed by multiplying man-months by 19, the standard number of actual workdays in a month (Boehm, 1981). The schedule estimation must be calculated using man-months for effort to determine the number of months and then multiply the months by 19. The equations below depict an example of the COCOMO long hand calculation results utilizing the baseline data.

The printout below, REPORT.OUT, shows a typical output from the single project environment. Estimated man-

Example 1: Basic COCOMO

$$MD = 2.4(64)^{1.05} \times 19$$
$$= 189.1 \times 19$$
$$= 3593$$

$$TDEV = 2.5(189.1)^{0.38} \times 19$$
$$= 18.3 \times 19$$
$$= 348$$

days is the value calculated using the COCOMO algorithms for effort necessary to complete the project. The actual man-days is the value calculated by the dynamic simulation model. This value represents what the actual effort would be to complete

the project after all of the project management policies were incorporated into the COCOMO estimate. Estimated Schedule is the value calculated using the COCOMO algorithms for the time in days it takes to complete the project. The actual schedule is the value calculated by the dynamic simulation model. This value represents the actual time it would take to complete the

**********REPORT.OUT**********

Estimated man-days	Actual man-days	Percent Error	Estimated Schedule	Actual Schedule	Percent Error
3592.97	3590.89	0.00	348.21	349.00	0.00

**** This data is available in REPORT.OUT ****
**** Each time the model is run REPORT.OUT will change ****

****

project after all of the project management policies were incorporated into the COCOMO estimate.

Comparing the long hand calculations above with the results generated by the combined model in REPORT.OUT proves that the algorithms within the model provide the simulation with the correct estimates.

b. Intermediate COCOMO

The intermediate model works on the same general principles as the basic model. The equations for the schedule calculations are identical. The differences in the two models are reflected in the differences between the equations for

effort estimation. First, the coefficients are different. The coefficients in the intermediate model must account for the aggregate effect of the effort multipliers (Boehm, 1981, p. 117). There are 15 effort multipliers or cost drivers which are multiplied by one another to determine the effort adjustment factor (EAF). The Intermediate COCOMO equations are displayed in Table 4-2.

Chapter III described how the values for each cost driver is reached. The EAF is incorporated into the effort equation through direct multiplication. The equations below show an example of how EAF is calculated and applied to the effort equation in the intermediate COCOMO model.

TABLE 4-2: INTERMEDIATE COCOMO EQUATIONS

Intermediate COCOMO Equations

Mode	Effort	Schedule
Organic	$MM = 3.2(KDSI)^{1.05} \times EAF$	$TDEV = 2.5(MM)^{0.38}$
Semidetached	$MM = 3.0(KDSI)^{1.12} \times EAF$	$TDEV = 2.5(MM)^{0.35}$
Embedded	$MM = 2.8(KDSI)^{1.20} \times EAF$	$TDEV = 2.5(MM)^{0.32}$

EAF - Effort Adjustment Factor ($CD_1 \cdot CD_2 \cdot CD_3 \ldots \cdot CD_{15}$)
where CD is one of 15 Cost Drivers

The cost drivers tend to effect the overall effort cost the same (i.e., the cost either increases or decrease as a cost driver rating goes from very low to very high). For example, if you rate the software reliability of a project

Example 2: Intermediate COCOMO

$$EAF = 0.75 \times 1.06 \times 1.08 \times 1.00 \ldots\ldots \times 1.00 = 0.8586$$

$$MD = 3.2(64)^{1.05} \times 19 \times 0.8586$$
$$= 252.1 \times 19 \times 0.8586$$
$$= 4113$$

$$MM = 252.1 \times \frac{0.8586}{1.08} = 200.4$$

$$TDEV = 2.5(200.4)^{0.38} \times 19 \times 0.85$$
$$= 18.7 \times 19 \times 0.85$$
$$= 302$$

very low it will cut the effort cost by 25%, but if you rate the programmers capability very low it will increase the cost of the project by 42%. This is true for all the cost drivers except for the required development schedule cost driver. As the rating goes from low to high, the effort multiplier decreases until the rating becomes nominal (1.00), and it increases as the ratings become higher. Intuitively, this is true because it will take more effort to either compress or expand the work schedule. The schedule cost driver does not affect the algorithm for determining the effort but does affect the algorithm to determine schedule. If you increase the effort by compressing or expanding the schedule and use

that calculated value of effort in the schedule equation, the schedule will always increase in time. This is not true if you have increased effort to compress the schedule. Therefore, to calculate the schedule you must divide the effort by the schedule cost driver (also divide by 19 if working in man-days), and insert the adjusted effort in the schedule equation. Then multiply that number by the associated percentage for each rating level. For example, the effort multiplier for low is 1.08 and its associated percentage 85%. Multiplying by .85 would then account for a 15% compression in schedule. You must also multiply by 19 to determine schedule in days. The equations in the above example represent long hand calculation results using intermediate COCOMO. Comparing these results to those shown from the computer results in REPORT.OUT below, proves that the intermediate COCOMO algorithms within the model provide the simulation with correct and accurate estimates.

**********REPORT.OUT**********

Estimated man-days	Actual man-days	Percent Error	Estimated Schedule	Actual Schedule	Percent Error
4113.13	4108.86	0.00	301.60	305.00	0.01

**** This data is available in REPORT.OUT ****
**** Each time the model is run REPORT.OUT will change ****

****

2. **Test 2**

The following test was run with baseline data to prove the accuracy of the front-end portion of this model in the two project environment. The data used for this test is displayed in Figure 4-1 and is the same data used in the previous test. As in the first test, most of the variables are direct inputs to the simulation and calculations are not required. This test for the two project environment was conducted using the same input data for both projects. The projects were also treated as independent projects using Basic COCOMO. With no interaction between the projects, the results of each project in this test should have been identical with the test results from the project run in the single project environment. Comparing the results from the equations in Example 1 above to those shown from the computer results in REPORT.OUT below, proves that the COCOMO algorithms within the model provide the simulation with correct and accurate estimates.

**********REPORT.OUT**********

TOTMD1	CUMMD1	PERCENT ERROR	TDEV1	TIME1	PERCENT ERROR	PRODUCTIVITY OLD	NEW
3593	3591	0.00	348	348	0.00	59.89	59.92

TOTMD2	CUMMD2	PERCENT ERROR	TDEV2	T1ME2	PERCENT ERROR	PRODUCTIVITY OLD	NEW
3593	3591	0.00	348	348	0.00	59.89	59.92

**** This data is available in REPORT.OUT *****
**** Each time the model is run REPORT.OUT will change ****

****

3. Test 3

The following test was run with baseline data to prove the accuracy of the nominal productivity algorithms for input to the simulation model within the front-end portion of this model. This test was run in conjunction with Test 2. The results for nominal productivity are displayed as part of the output data in the REPORT.OUT file shown above.

There were two sets of algorithmic calculations that required testing. The first part of the test was to prove that the initial algorithms were correct, and the second part of the test was to prove that the nominal productivity dynamically updates as the variables associated with it are updated. The following set of equations are required to determine nominal productivity:

Nominal Productivity Equations

$$DP = (1 - (\%MDT + \%MDQA + \%MDR)) \times MD$$

ADP - Actual Development Productivity
DP - Development Productivity
NP - Nominal Productivity
MD - Man-days
D-MD - Development Man-days
%MDT - % MD for Tests
%MDQA - % MD for QA
%MDR - % MD for Rework

$$ADP = \frac{Size(LOC)}{D\text{-}MD}$$

$$\text{Staff Size} = \frac{MD/19}{TDEV/19}$$

Staff Size is entered into Table 4-2 to get Communication Overhead

$$NP = \frac{ADP}{0.6 \times (1 - \text{Comm overhesd})}$$

Table 4-3, displayed below, is necessary to retrieve the value for communication overhead from the average staff size input. In many cases, you must interpolate for proper communication overhead values.

TABLE 4-3: STAFF SIZE VS. COMMUNICATION OVERHEAD

Staff Size	Communication Overhead
0	0
5	0.015
10	0.06
15	0.135
20	0.24
25	0.375
30	0.54

NOTE: 30 applies to greater than 30 as well.

Nominal productivity is affected by many variables. The first variables that you need are inputs for percent of man-days for tests, percent of man-days for quality assurance (QA) and percent of man-days for rework. All three of these variables are critical to the success of the project, but are not considered an input to the process.

> The development of software systems involves a series of production activities where opportunities for injection of human falliabilities are enormous. Errors may begin to occur at the very inception of the process where the objectives...may be erroneously or imperfectly specified, as well as [errors that occur in] later design and development stages....Because of human inability to perform and communicate with perfection, software development is accompanied by a quality assurance activity (Pressman, 1987, p. 467).

Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, design, and coding. Testing in some instances is equivalent to 40 percent of the total project effort. Rework is the third critical element utilizing a percentage of project effort to correct errors located by quality assurance and testing (Pressman, 1987). These three variables are entered directly into the model. They are used for determining Development Productivity in man-days as shown.

DP = (1 - (0.22 + 0.11 + 0.14) x 3593

DP = 1904 man-days

Development Productivity is divided into the size of the project for the determination of the Actual Development Productivity. This productivity is defined as the outputs produced by the process divided by the inputs consumed by the process(Boehm, 1981, pp. 44). Since a certain percentage of effort must be expended on testing, quality assurance, and rework, it makes sense that Actual Development Productivity only account for the effort expended on the actual development.

ADP = 64000/1904 = 33.61 DSI/man-day

The next step is to divide the effort in man-months by the schedule in months, which provides the average staff size for the project. Entering this value into Table 4-2 you will

Staff Size = 189.1/18.3 = 10.3

be able to extract the Communication Overhead. The larger the staff size the more communication problems and breakdowns a project will incur. Therefore, a percentage of the communication overhead also affects productivity. Nominal productivity is determined through the division of the Actual Development Productivity by the multiplying the communication overhead percent by 0.6. It has been determined that approximately 60 percent of a single work day is utilized for technical development, which accounts for the 0.6 factor applied in the equation (Ghezzi, 1991, pp. 420).

$$NP = 33.61/[0.6 \times (1 - 0.0645)]$$

$$NP = 59.88$$

Nominal productivity consists of many variables which continually change values throughout the process of this model. As the effort changes, the schedule changes, and as the schedule or the effort change, the staff size changes, which causes the communication overhead to change. The equations listed below shows, step by step, how nominal productivity changes dynamically with changes in the effort and schedule variables. Comparing the new and old productivity values from the computer results in REPORT.OUT from Test 3 to the results from the above equations proves the algorithms in the model provide the simulation with the correct nominal productivity, and that Nominal Productivity is a dynamically changing variable throughout the simulation process.

Dynamic change in NP

DP = (1 - (0.22 + 0.11 + 0.14) x 3591

DP = 1903

ADP = 33.63

$$\text{Staff Size} = \frac{189}{18.3} = 10.3$$

$$NP = \frac{33.63}{0.6 * (0.9355)} = 59.91$$

4. Experiment

This experiment was conducted to test the sensitivity of just one of the many applications and advantages of utilizing a coupled modeling system. The same data was used for both projects as in Test 3 above.

Assuming the original COCOMO inputs for cost and schedule were accurate, the simulation model ran in the two project environment with a work force ceiling of 15 people. To run this experiment, there were several adjustments that were made to both models. The simulation model was adjusted to allow for interaction between the two projects, as not to run independently of each other as in Test 3. The other model was adjusted to include the ability to enter the work force ceiling variable and the start dates for both projects.

The work force ceiling is an attribute of the average staff size. For both of these projects in this experiment, the average staff size was approximately 10. Therefore, on the average there would be only 20 people working on both projects at any single point in time. The work force ceiling of a project is the total number of people that management will allow to work on a project at any particular time.

In many software companies resources are shared between projects. This was one of the themes in the experiment. By reducing the work force ceiling to 25 percent of the combined total average staff size, or 15, the simulation is forced to create an environment of priorities which would ideally return the minimum effort required to complete both projects in the shortest and most effective time frame.

The iterative loop process in this model compares effort estimated to effort actual and schedule estimated to schedule actual. Prior to beginning the initial simulation run, the user must provide inputs for the desired accuracy levels between the estimates and the actual results for effort and schedule. For example, inputting 0.05 would set the effort error level of the estimate to 5% of the actual, where the error rate[1] is as follows:

---

[1] For this study the standard format for Error Rate is the absolute value [ABS(x)] of the Estimate minus the Actual divided by the Actual.

Error Rate = ABS(Estimate - Actual) / Actual

It also requires an input to restrict the number of iterations. This would prevent the possibility of an endless loop if the model could not reach the accuracy level requested. After the conclusion of each simulation run, the iterative loop process evaluates the results to determine if they meet user requirements for accuracy. If not, then the effort, schedule and productivity variables are adjusted and re-entered into the simulation as new estimates. If they are, then the program is terminated and the results are saved in REPORT.OUT.

The start date variable for each project was included to allow for different start dates of projects. This can determine how one project in a later phase of project development is affected by another project, just beginning, and are required to share resources.

For this experiment, the work force ceiling used was 15. This experiment was run four times changing the start dates each time. Project 1 always started at time zero, while project 2 was zero for the first run, 100 for the second, 200 for the third and 300 for the last run. There are several ways the results are displayed. The desired accuracy level used was 1 percent for effort and 1 percent for schedule. For the experiment run with the same start dates, Figures 4-2 and 4-3 show the Effort vs. number of Iterations of actual and estimated effort from both projects and the Schedule vs.

number of Iterations of actual and estimated schedules from both projects, respectfully.

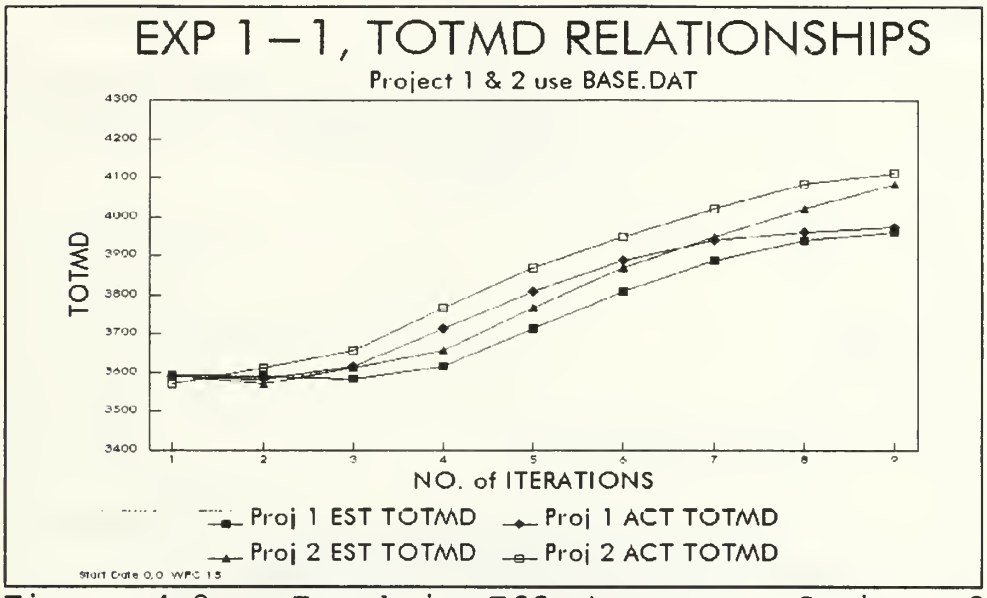

Figure 4-2: Trend in Effort over a Series of Iterations

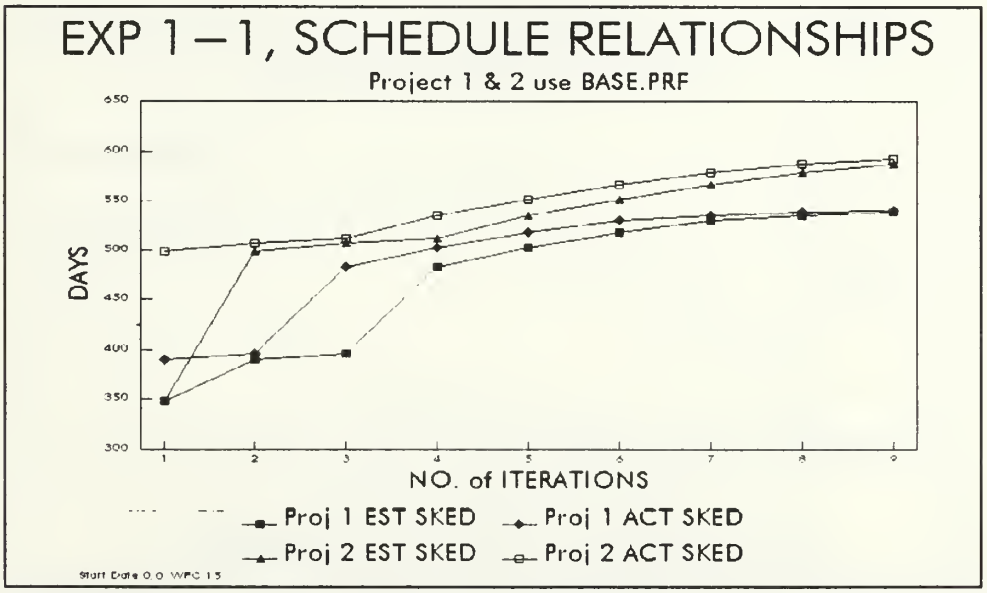

Figure 4-3: Schedule Trends over a Series of Iterations

Figures 4-4 and 4-5 represent the same comparisons as above; however, the start dates are no longer the same in this test. The start date in the second project is now 100 days later than the first.

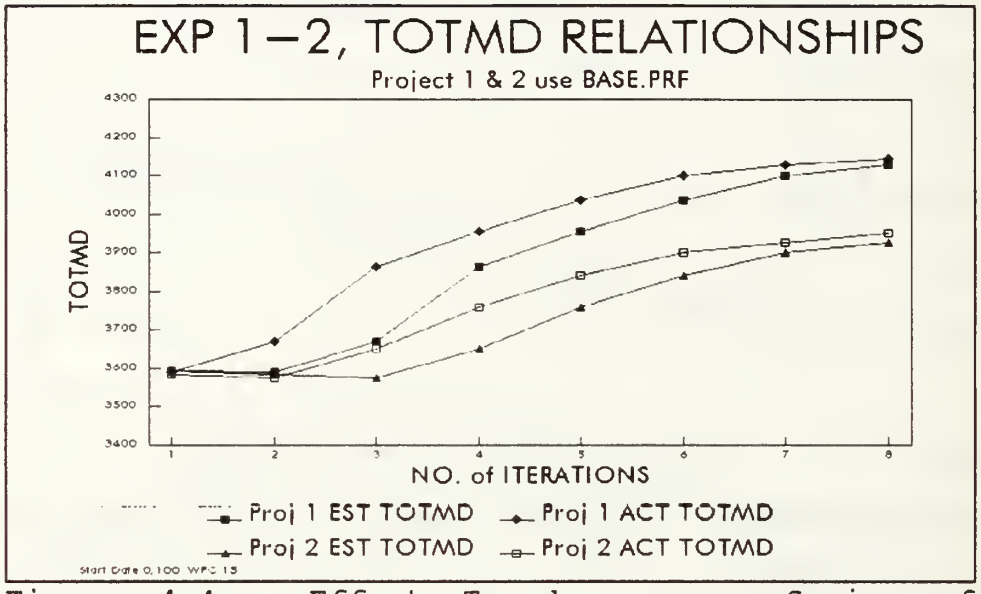

Figure 4-4: Effort Trends over a Series of Iterations with Different Start Dates

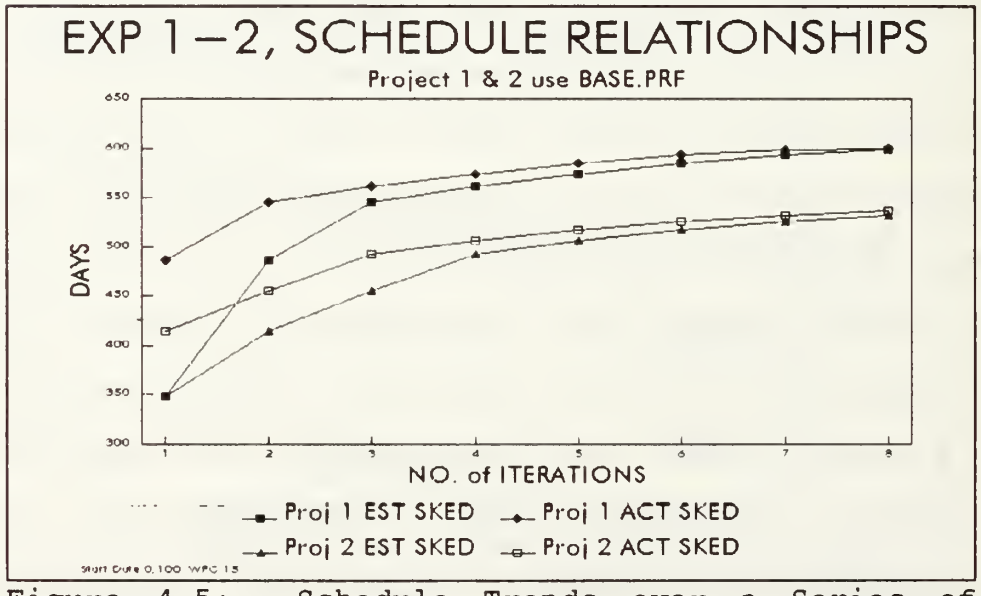

Figure 4-5: Schedule Trends over a Series of Iterations with Different Start Dates

Figure 4-6 and 4-7 show how the percent error changes over the number of iterations for the effort and schedule of both projects. Both projects of Figure 4-6 had the same start dates. Figure 4-7 had the second project start 100 days after the first.

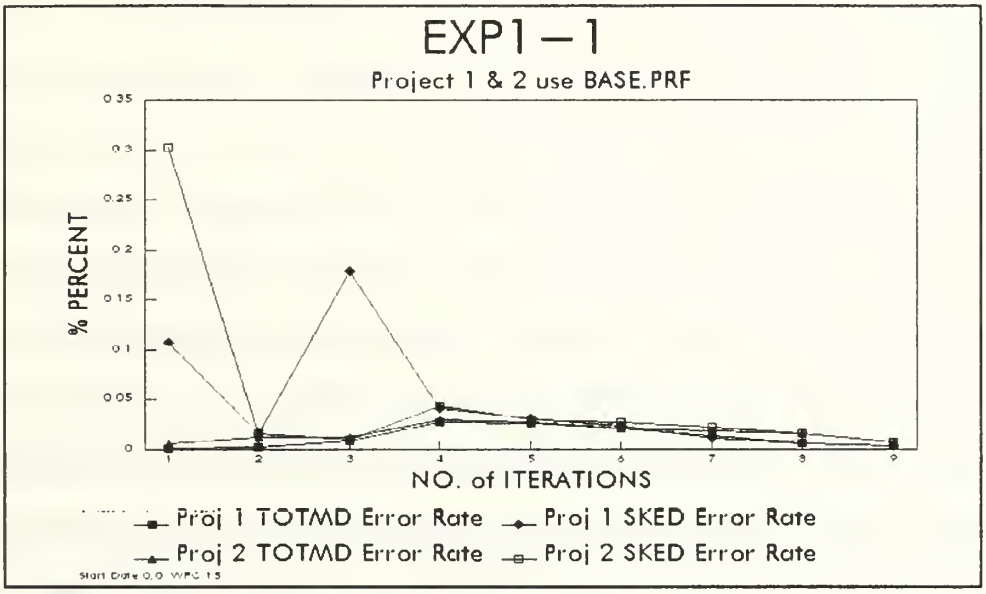

Figure 4-6: Error Rate Trends

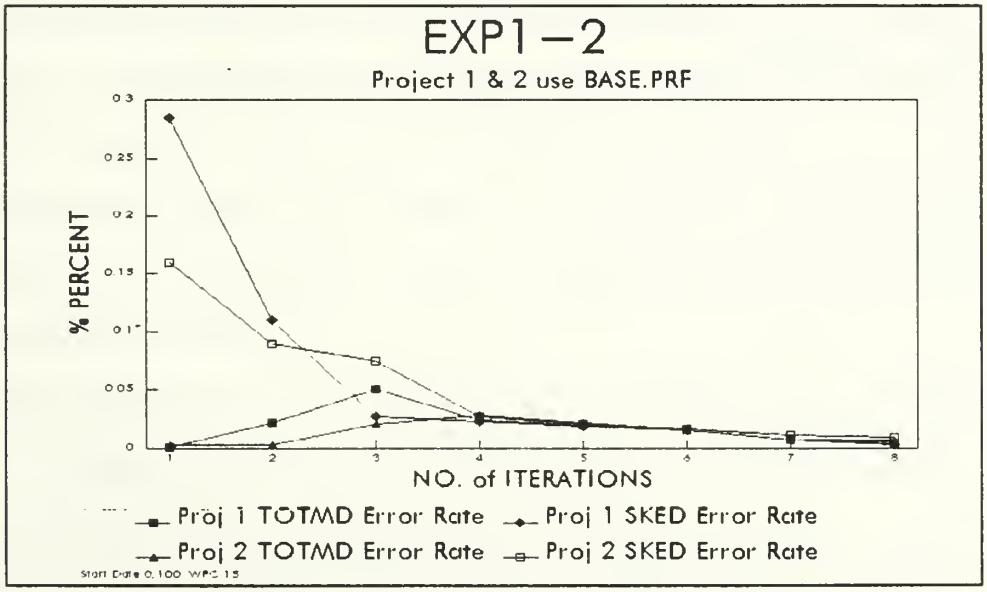

Figure 4-7: Error Rate Trends with Different Start Dates

The third and fourth run of this test incremented project 2 start dates by 100 days per run. In both cases, for this size project, sharing of resources with the second project start date 200 days or later had no significant effect on either project. There are many distinguishable trends that can be identified from the above graphs. Error Rate analysis shows that as the number of iterations are increased, the better the percent error rate. The error rate stabilized after four iterations and continued to approach zero. The effort and schedule related graphs show an expected increase in costs due to the sharing of resources. However, in all cases, even with the variable start dates, the estimated and actual costs tended to approach one another over the number of iterations. In addition, there was also a tendency for the costs to plateau or level off. This could lead to providing an upper cost limit on software development costs. Chapter V will discuss these issues in greater detail.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The primary objective of this thesis was to investigate the utility of coupling the COCOMO model with a Systems Dynamics Model of Software Project Management. It was expected that a combined model would allow for a richer and more complete set of cost drivers, thus increasing software cost estimation accuracy. The premise was coupling a model that quantified estimation based solely on objective variables, with a dynamic simulation model, which incorporates subjectivity into project management issues. This coupling would create an environment which not only addresses the necessary objective variables but also addresses many subjective variables of project management that tend to have an enormous impact on the cost and schedule estimates of a project. It was also expected that a coupled model would provide a means for more extensive sensitivity analysis. Another main objective of this thesis was to investigate the opportunity to optimize cost estimation procedures in a two project environment.

As a first step a simple "C" program was designed for basic COCOMO to ensure that an interface could be accomplished, i.e., an algorithmic model could be coupled with

a dynamic simulation model. After a successful initial test, four additional "C" programs were designed as well as expanding the first to include the ability to choose between the basic or intermediate COCOMO models. The programs are explained in detail in Chapter II. There were two separate environments incorporated into the design.

The single project environment was developed to study independent projects. The results of the tests in Chapter IV proved that the COCOMO algorithms are correct in both the basic and intermediate COCOMO models. The development of this type of system lends itself to the ease of variable entry. Whether the variable is from COCOMO or the simulation model, it is an ideal environment for studying the sensitivity of results to changes of a single variable on the entire project.

The two project environment was developed to incorporate the use of sensitivity analysis to investigate the opportunity to optimize cost estimation in an environment of shared resources. The experiment conducted in Chapter IV established the work force as the shared resource and limiting factor. The experiment was run four times with the start date of the second project being adjusted by 100 days each run. For this experiment, the results were quite conclusive that over a series of iterations the adjustments in a shared resource environment cost estimation and schedule estimates could be refined. This is apparent from the graphs of Chapter IV. Comparison of error rates clearly shows that after four

iterations the error rate not only remained below five percent, but continued to decrease after each additional iteration toward an optimal zero percent error. In the other two series of graphs, the displays depict the estimates vs. actual results. In all cases, after several iterations, there is a trend of convergence between estimate and actual costs. This trend, however, does not occur without additional cost in effort and schedule. Initially, the additional costs were fairly significant, but as the number of iterations increased, the results not only converged but they also tended to level off. This could give a project manager a much more realistic estimate of project cost and also provide top management with fairly realistic high-low estimate of total project cost.

The experiment also examined the sensitivity of start dates in the two project environment. With the remaining variables constant, the start date of project 2 was adjusted to 100 days after the start date of project 1. Two additional runs under the same conditions were also conducted. Each additional run increased the start date of project 2 by 100. Therefore, the fourth run was to demonstrate the effects of sharing resources with a project starting 300 days after the start of another project. The results indicate that the cost of sharing resources is less for two projects starting the same date than for two projects starting 100 days apart. The impact seemed to be greater on the first project because of the need to reallocate less resources over the same work

requirements. Another interesting result was that after approximately two thirds of project 1 was completed, sharing resources no longer had a significant impact on either project. These results conclude that it is easier to manage a project with a known set of resources from the beginning rather than having to shift personnel and priorities one third of the way through a project. This is just one example of how a coupled model can utilize sensitivity analysis to study and understand the many aspects and variables associated with Software Project Management.

Using COCOMO and defining its objective variables provide cost estimations which tend to fall short by itself. By studying and learning how the subjective variables affect the system can lead to not just an improved understanding of project management, but to the ability of narrowing the percent error in cost estimation currently plaguing the software industry. For example, COCOMO is used to provide initial estimates to the simulation model. COCOMO has only been accurate to a point. With respect to historical model accuracy, the Basic COCOMO estimates were within a factor of 1.3 of the actual cost only 29% of the time, and within a factor of 2 of the actual cost only 60% of the time (Boehm, 1981, p. 114). Applied to this experiment, that would infer that the actual project results would have been between 4670 and 7186 man-days for effort only 29% and 60% of the time, respectively. From the results in chapter four, this coupled

model, utilizing an iterative loop process, continually addressed subjective variables not incorporated into COCOMO to establish what appears to be very accurate results. In essence the coupled model accepts COCOMO estimates and then incorporates the subjective influences of project management to eliminate the need for a "safety factor" to account for the factors 1.3 to 2 of the actual produced by COCOMO. This experiment produced effort estimates within 13% of COCOMOs original estimates and schedule estimates within 41% of COCOMO estimates. The significant difference in schedule is due the sharing of resources.

B. RECOMMENDATIONS FOR FUTURE RESEARCH

Several areas are available for conducting follow-on and future research. Several prominent topics are: (1) refining the current system, (2) conducting experiments using the current system to evaluate the variables most sensitive to project development to define a more complete set of cost drivers, and (3) utilizing the model in an actual project environment to study whether the system will optimize cost estimation in a two project environment.

1. Refining the Current System

Since one of the purposes of this thesis was to optimize cost estimation in a two project environment, the programs developed were focused on the ability to accomplish this. A possible follow-on research would be to enhance the

current system with the ability to change variables between iterations, thus giving the user greater flexibility and control in conducting experiments.

2. **Use of Current System for Sensitivity Analysis Experiments**

This thesis was developed to create an avenue to study and establish a richer set of cost drivers. The tool now exists to use sensitivity analysis to determine the variables that are most sensitive to project development. This would be an obvious next step for a follow-on thesis topic.

3. **Determine Real World Advantages**

Laboratory experiments and use of historical data can lead to very conclusive results. However, since this model contains many subjective variables, among the countless number of actual variables which effect Software Project Development, the proof of the usefulness and effectiveness of this system cannot be determined until it is used by industry. This is the ideal experiment and test for follow-on studies.

APPENDIX A

```
/* ********************************************************** */
/* * Author: Richard W. Smith    Advisor:  Prof. Abdel-Hamid  * */
/* * Program: Main               Lang: C                      * */
/* * Used Shareware <windows.h> in project environment        * */
/* ********************************************************** */

/* This is one of 5 programs written and interfaced with the */
/* Dynamic Simulation Model.  This particular program is a simple */
/* program that allows the user the ability to select the single */
/* project environment or the two project environment from a menu. */
/* Once the user selects this program is terminated. */

/* The following headers were used and needed to utilize the */
/* library functions used throughout this program. */

#include <windows.h>
#include <stdio.h>

/* Declarations for the menu windows boarder and background */

int bat;                          /* border atrib */
int wat;                          /* window atrib */

/* The following are static structures developed to be */
/* used throughout the program in pop-up menus for various */
/* user selection requirements.  The learning curve for */
/* the use of windows.h was considerable, however, once */
/* learned it is fairly simple to create menus. */

static struct pmenu intelc50 =
 {0, FALSE, 0,
 2, 3,
 1, 8, "     INITIALIZATION OF PROGRAMS", 0,
 2, 8, "    (Select one of the following)", 0,
 4, 12, "1 - SINGLE Project Environment", 1,
 5, 12, "2 - TWO Project Environment", 2,
 7, 3, "Select environment you wish to use and press enter:",0,
99, 99, "",99
 };


WINDOWPTR w3;                             /* window to use */

void main()
{
     int sel;

  /* bat is the boarder attribute for the pop-up window */
  /* sets background to blue and boarder to white */
     bat = v_setatr(BLUE,WHITE,0,0);
  /* wat is the window attribute for the pop-up window */
  /* sets background to blue and text to white */
```

```
    wat = v_setatr(BLUE,WHITE,0,0);

  /* Introduction window is declared as w3 above and */
  /* is opened and closed as if it were a file */

  clrscr();
  wn_init();
  w3 = wn_open(0,5,10,60,12,wat,bat);
  if(!w3) exit(1);

  wn_printf(w3,"    ************************************************
                  \n\n");
  wn_printf(w3,"                Welcome to the coupling of \n");
  wn_printf(w3,"           COCOMO and a SYSTEMS DYNAMICS MODEL\n\n");
  wn_printf(w3,"    ************************************************
                  \n\n");
  wn_printf(w3,"          Programmed by Richard W. Smith & Tarek
                  Abdel-Hamid\n\n\n");
  wn_printf(w3,"                Press [ANY KEY] to continue...");

  v_getch();  /* Stops the program and awaits any keyboard entry */
  wn_close(w3);

  clrscr();
  /* Sets the pop-up window size and assigns a ststic structure */
  /* for menu operation */
  /* sel awaits an appropriate keyboard entry from the menu choices */

  sel = wn_popup(0, 5, 10, 55, 10, wat, bat, &intelc50, TRUE);

  switch (sel)   /* case statement to direct remainder of coupled */
                 /* system */
  {
      case 1:
         exit (0);  /* if selected, program exits to DOS for */
                    /* system call to INPUT1.EXE */
      case 2:
         exit (1);  /* if selected, program exits to DOS for */
                    /* system call to INPUT2.EXE */

  }
}  /* end program */
```

APPENDIX B

```
/* ********************************************************** */
/* * Author: Richard W. Smith    Advisor:  Prof. Abdel-Hamid  * */
/* * Program: Main               Lang: C                      * */
/* * Used Shareware <windows.h> in project environment        * */
/* ********************************************************** */

/* This is one of 5 programs written and interfaced with the */
/* Dynamic Simulation Model.  This particular program completes */
/* two tasks.  First it accepts input variables for the dynamic */
/* simulation model and COCOMO acting as a front end for the  */
/* model in the single project environment.  Then it makes all */
/* the necessary COCOMO calcuations for either the Basic or  */
/* the intermediate versions of COCOMO. */


/* The following headers were used and needed to utilize the */
/* library functions used throughout this program. */

#include <windows.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <dir.h>
#include <string.h>

/* Prototypes for the functions which will be */
/* described below.                           */

int filelist(void);
void  model_in(float  *,float  *,int  *,float  *,int,char  a[],float  *,float
*,int,float);
void icocomo_in(float *,float *,int,char *);
void file_save(float *,float *,float *,char *,float,int *,int,float);
void file_prnt(float,float,int *,float *);
float interp(float);
float prod(float *,float,float,int *);
void calc(float *,int *,float *,float *,float,float,float);
void initial(float *);

/* Declarations for the menu windows boarder and background */

int bat;                        /* border atribute */
int wat;                        /* window atribute */

/*Pointer to file being used*/

FILE * textfile;
FILE * fin;
FILE * fout;
FILE * fnew;

WINDOWPTR w3;                   /* window declaration */
WINDOWPTR w4;                   /* window declaration */
```

```
/* The following are static structures developed to be */
/* used throughout the program in pop-up menus for various */
/* user selection requirements.  The learning curve for */
/* the use of windows.h was considerable, however, once */
/* learned it is fairly simple to create menus. */

static struct pmenu intelc =
 {0, FALSE, 0,              /* Must be FALSE */
 1, 3,                      /* The 1 initiates which row */
                        /* The 3 determines number of lines */
                        /* which can be highlighted after row */
/* row,col */
 1, 20, "INITIAL MENU", 0,
 4, 12, "1 - LOAD project from disk.", 1,
 5, 12, "2 - NEW project.", 2,
 6, 12, "3 - EXIT Program.", 3,
 9, 3, "Select with number or cursor and press [ENTER]...",0,
99, 99, "",99
 };

static struct pmenu intelc23 =
 {0, FALSE, 0,
 1, 3,
 1, 20, "NEW PROJECT MENU", 0,
 4, 12, "1 - Display/Edit.", 1,
 5, 12, "2 - RUN Dynamic Simulation.", 2,
 6, 12, "3 - QUIT menu.", 3,
 9, 3, "Select with number or cursor and press [ENTER]...",0,
99, 99, "",99
 };

static struct pmenu intelc0 =
 {0, FALSE, 0,
 1, 4,
 1, 21, "   MAIN MENU", 0,
 3, 15, "1 - LIST projects on disk.", 1,
 4, 15, "2 - SELECT desired project.", 2,
 5, 15, "3 - RUN Dynamic Simulation.",3,
 6, 15, "4 - QUIT menu.", 4,
 9, 3, "Select with number or cursor and press [ENTER]...",0,
99, 99, "",99
 };


static struct pmenu intelc19 =
 {0, FALSE, 0,
 2, 3,
 1, 15, "       COCOMO MODEL", 0,
 2, 15, "        ESC - EXIT ", 0,
 4, 15, "1 - Basic COCOMO Model", 1,
 5, 15, "2 - Intermediate COCOMO Model", 2,
 7, 3, "Select the model you wish to use and press enter:",0,
99, 99, "",99
 };

static struct pmenu intelc21 =
 {0, FALSE, 0,
 2, 3,
 1, 6, "Current data file is for the Basic COCOMO", 0,
 2, 6, "    (Select one of the following)", 0,
 4, 10, "1 - CONTINUE Basic COCOMO Model", 1,
```

```
 5, 10, "2 - Intermediate COCOMO Model", 2,
 7, 3, "Select the model you wish to use and press enter:",0,
99, 99, "",99
 };


static struct pmenu intelc22 =
 {0, FALSE, 0,
 2, 3,
 1, 16, "SAVING FILES", 0,
 2, 6, "    (Select one of the following)", 0,
 4, 10, "1 - SAVE changes under Same Name ", 1,
 5, 10, "2 - SAVE changes under New Name", 2,
 7, 3, "Select the model you wish to use and press enter:",0,
99, 99, "",99
 };


static struct pmenu intelc20 =
 {0, FALSE, 0,
 2, 4,
 1, 10, "      COCOMO MODE SELECTION", 0,
 2, 10, "           ESC - EXIT ", 0,
 4, 18, "1 - Organic", 1,
 5, 18, "2 - Semi-detached", 2,
 6, 18, "3 - Embedded",3,
 8, 3, "Select the appropriate mode and press enter:  ",0,
99, 99, "",99
 };


static struct pmenu intelc1 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "       RELY(Required software reliability)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 0.75", 1,
 5, 15, "2 - Low; 0.88", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 1.15",4,
 8, 15, "5 - Very High; 1.40", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };



static struct pmenu intelc2 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "          DATA(Database size):", 0,
 2, 2, "               ESC - EXIT ", 0,
 4, 15, "1 - Low; 0.94", 1,
 5, 15, "2 - Nominal; 1.00", 2,
 6, 15, "3 - High; 1.08", 3,
 7, 15, "4 - Very High; 1.16",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };
```

```
static struct pmenu intelc3 =
 {0, FALSE, 0,
 2, 7,
 1, 2, "           CPLX(Product complexity)", 0,
 2, 2, "                 ESC - EXIT ", 0,
 4, 15, "1-Very Low; 0.70",1,
 5, 15, "2-Low; 0.85",2,
 6, 15, "3-Nominal; 1.00", 3,
 7, 15, "4-High; 1.15",4,
 8, 15, "5-Very High; 1.30", 5,
 9, 15, "6-Extra High; 1.65", 6,
 11, 1, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };


static struct pmenu intelc4 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "         TIME(Exection time constraint)", 0,
 2, 2, "                 ESC - EXIT ", 0,
 4, 15, "1 - Nominal; 1.00", 1,
 5, 15, "2 - High; 1.11", 2,
 6, 15, "3 - Very High; 1.30", 3,
 7, 15, "4 - Extra High; 1.66",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc5 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "         STOR(Main storage constraint)", 0,
 2, 2, "                 ESC - EXIT ", 0,
 4, 15, "1 - Nominal; 1.00", 1,
 5, 15, "2 - High; 1.06", 2,
 6, 15, "3 - Very High; 1.21", 3,
 7, 15, "4 - Extra High; 1.56",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc6 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "       VIRT(Virtual machine volatility)", 0,
 2, 2, "                 ESC - EXIT ", 0,
 4, 15, "1 - Low; 0.87", 1,
 5, 15, "2 - Nominal; 1.00", 2,
 6, 15, "3 - High; 1.15", 3,
 7, 15, "4 - Very-High; 1.30", 4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc7 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "         TURN(Computer turnaround time)", 0,
 2, 2, "                 ESC - EXIT ", 0,
 4, 15, "1 - Low; 0.87", 1,
```

```
 5, 15, "2 - Nominal; 1.00", 2,
 6, 15, "3 - High; 1.07", 3,
 7, 15, "4 - Very High; 1.15",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc8 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "           ACAP(Analyst capability)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.46", 1,
 5, 15, "2 - Low; 1.19", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.86",4,
 8, 15, "5 - Very High; 0.71", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc9 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "         AEXP(Applications experience)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.29", 1,
 5, 15, "2 - Low; 1.13", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.91",4,
 8, 15, "5 - Very High; 0.82", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc10 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "         PCAP(Programmer capability)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.42", 1,
 5, 15, "2 - Low; 1.17", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.86",4,
 8, 15, "5 - Very High; 0.70", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc11 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "       VEXP(Virtual machine experience)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.21", 1,
 5, 15, "2 - Low; 1.10", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.90",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };
```

```
static struct pmenu intelc12 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "      LEXP(Programming Language experience)", 0,
 2, 2, "                    ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.14", 1,
 5, 15, "2 - Low; 1.07", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.95",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc13 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "       MODP(Use of modern programming practices)", 0,
 2, 2, "                      ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.24", 1,
 5, 15, "2 - Low; 1.10", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.91", 4,
 8, 15, "5 - Very High; 0.82", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc14 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "           TOOL(Use of software tools)", 0,
 2, 2, "                    ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.24", 1,
 5, 15, "2 - Low; 1.10", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.91", 4,
 8, 15, "5 - Very High; 0.83", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc15 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "      SCED(Required development schedule", 0,
 2, 2, "                   ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.23", 1,
 5, 15, "2 - Low; 1.08", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 1.04",4,
 8, 15, "5 - Very High; 1.10", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
};


/* Function which lists all the data files (*.PRF) */
/* in the current directory.  */

int filelist(void)
{
```

```
    struct ffblk ffblk;
    int done4;
    printf("Data file listing: \n\n");          /* prf for profile */
    done4 = findfirst("*.prf",&ffblk,0);  /* finds first .prf file */
    while(!done4)
    {
         printf("   %s\n",ffblk.ff_name);
         done4 = findnext(&ffblk);          /* finds the next .prf file */
    }
    return(1);
}

/* This function accepts numerous pointers to various strings */
/* which allows the user to select variables from the display */
/* and change the value of current simulation input variables. */

void model_in(float *fptr,float *PCNT,int *KDSI,float *results,int mode,char
fname1[],float *EAF1,float *cdrate,int done1, float mf1)
{

    /* Declarations for this function */
    int choice1, choice2, choice3;
    float exp1,exp2;
    char string1[] = "Organic";
    char string2[] = "Semi-detached";
    char string3[] = "Embedded";
    char string[14];

    switch(mode)  /* mode variable is passed in to function */
    {             /* used to display one of the 3 strings declared */
              /* above for display on this screen; switch/case format*/
    case 1:
         strcpy(string,string1);
         break;
    case 2:
         strcpy(string,string2);
         break;
    case 3:
         strcpy(string,string3);
         break;
    }

    /* clears screen and displays variables on screen in below format */

    while(!done1)
    {
     clrscr();
     printf("                ****************************************\n");
     printf("                      MODEL INPUTS for %s       \n",fname1);
       printf("                           %s Mode\n",string);
     printf("                ****************************************\n\n");
       printf("                1. INUDST:  %5.3f              8.  (1) TPFMQA[1]:
%5.3f\n",fptr[0],fptr[12]);
     printf("               2. ADMPPS:  %5.3f                   (2) TPFMQA[2]:
%5.3f\n",fptr[1],fptr[13]);
     printf("               3. HIREDY:  %5.3f                   (3) TPFMQA[3]:
%5.3f\n",fptr[2],fptr[14]);
     printf("                4. AVEMPT:  %5.3f                  (4) TPFMQA[4]:
%5.3f\n",fptr[3],fptr[15]);
     printf("               5. TRPNHR:  %5.3f                   (5) TPFMQA[5]:
%5.3f\n",fptr[4],fptr[16]);
```

```
    printf("                6. ASIMDY:   %5.3f                    (6) TPFMQA[6]:
%5.3f\n",fptr[5],fptr[17]);
    printf("                 7.  (1) TNERPK[1]:   %5.3f           (7) TPFMQA[7]:
%5.3f\n",fptr[6],fptr[18]);
    printf("                     (2) TNERPK[2]:   %5.3f           (8) TPFMQA[8]:
%5.3f\n",fptr[7],fptr[19]);
    printf("                     (3) TNERPK[3]:   %5.3f           (9) TPFMQA[9]:
%5.3f\n",fptr[8],fptr[20]);
    printf("                    (4) TNERPK[4]:   %5.3f          (10) TPFMQA[10]:
%5.3f\n",fptr[9],fptr[21]);
    printf("                         (5) TNERPK[5]:   %5.3f        9.   DEVPRT:
%5.3f\n",fptr[10],fptr[22]);
    printf("                         (6) TNERPK[6]:   %5.3f       10.   DSIPTK:
%5.3f\n\n",fptr[11],fptr[23]);
    printf("                   11. Size of project (KDSI):  %d\n\n",KDSI[0]);
    printf("                     12. EXIT and SAVE changes.\n\n");

   /* allows user to select a variable using assigned number and */
   /* change current value by displaying just the variable selected */
   /* once the new value is entered fuction returns to the display */
   /* screen for user to see changes and allow additional changes */

   printf("     Enter number of parameter you wish to change:  ");
   scanf("%d",&choice1);

   switch(choice1)
   {
        case 1:
               clrscr();
             gotoxy(10,10);
             printf("Enter Initial Under Staffing Level Factor:  ");
             scanf("%f",&fptr[0]);
             break;
        case 2:
               clrscr();
             gotoxy(10,10);
             printf("Enter Average Daily Manpower per Staff Member:  ");
             scanf("%f",&fptr[1]);
             break;
        case 3:
             clrscr();
             gotoxy(10,10);
             printf("Enter Hiring Delay:  ");
             scanf("%f",&fptr[2]);
             break;
        case 4:
               clrscr();
             gotoxy(10,10);
             printf("Enter Average Employment Time:  ");
             scanf("%f",&fptr[3]);
             break;
        case 5:
               clrscr();
             gotoxy(10,10);
             printf("Enter Training Overhead:  ");
             scanf("%f",&fptr[4]);
             break;
        case 6:
               clrscr();
             gotoxy(10,10);
             printf("Enter Average Assimilation Delay:  ");
```

```
            scanf("%f",&fptr[5]);
            break;


    /* The TNERPK has several entries for this one variable by using */
    /* a second set of values for each entry the user can change one */
    /* entry at a time vice entring all the values each time even if */
    /* one value needed to be changed.  */

        case 7:
            printf("     Enter subscript value of TNERPK parameter you wish to
change:  ");
            scanf("%d",&choice2);
            switch(choice2)
            {
                case 1:
                    clrscr();
                        gotoxy(10,10);
                    printf("Enter Error rate[1]: ");
                    scanf("%f",&fptr[6]);
                    break;
                case 2:
                    clrscr();
                        gotoxy(10,10);
                        printf("Enter Error rate[2]: ");
                    scanf("%f",&fptr[7]);
                    break;
                case 3:
                    clrscr();
                        gotoxy(10,10);
                        printf("Enter Error rate[3]: ");
                    scanf("%f",&fptr[8]);
                    break;
                case 4:
                    clrscr();
                        gotoxy(10,10);
                        printf("Enter Error rate[4]: ");
                    scanf("%f",&fptr[9]);
                    break;
                case 5:
                    clrscr();
                        gotoxy(10,10);
                        printf("Enter Error rate[5]: ");
                    scanf("%f",&fptr[10]);
                    break;
                case 6:
                    clrscr();
                        gotoxy(10,10);
                        printf("Enter Error rate[6]: ");
                    scanf("%f",&fptr[11]);
                    break;
                default:
                    break;
            }
                break;

            /* TPFMQA set-up same way as TNERPK for same reasons */

        case 8:
```

```
                printf("     Enter subscript value of TPFMQA parameter you wish
to change:  ");
                scanf("%d",&choice3);
                switch(choice3)
                {
                    case 1:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[1]:  ");
                        scanf("%f",&fptr[12]);
                        break;
                            case 2:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[2]:  ");
                        scanf("%f",&fptr[13]);
                        break;
                            case 3:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[3]:  ");
                        scanf("%f",&fptr[14]);
                        break;
                            case 4:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[4]:  ");
                        scanf("%f",&fptr[15]);
                        break;
                    case 5:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[5]:  ");
                        scanf("%f",&fptr[16]);
                        break;
                    case 6:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[6]:  ");
                        scanf("%f",&fptr[17]);
                        break;
                            case 7:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[7]:  ");
                        scanf("%f",&fptr[18]);
                        break;
                            case 8:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[8]:  ");
                        scanf("%f",&fptr[19]);
                        break;
                            case 9:
                        clrscr();
                            gotoxy(10,10);
                        printf("Enter Planned Fraction of Manpower for QA[9]:  ");
                        scanf("%f",&fptr[20]);
                        break;
                    case 10:
                        clrscr();
```

```
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[10]:  ");
                scanf("%f",&fptr[21]);
                break;
            default:
                break;

        }
            break;
        case 9:
                    clrscr();
            gotoxy(10,10);
            printf("Enter DEVPRT:  ");
            scanf("%f",&fptr[22]);
            break;
        case 10:
                    clrscr();
            gotoxy(10,10);
            printf("Enter  Nominal  Potential  Productivity  MD  percent  :
\n\n");
            printf("              Percent MD for tests:  ");
            scanf("%f",&PCNT[0]);
            printf("\n  Please be consistent with TPFMQA input values)\n");
            printf("              Percent MD for QA:     ");
            scanf("%f",&PCNT[1]);
            printf("              Percent MD for Rework: ");
            scanf("%f",&PCNT[2]);

            break;
        case 11:
            clrscr();
                    gotoxy(10,10);
            printf("Enter new Size of project (KDSI):  ");
            scanf("%d",&KDSI[0]);
            break;
        default:
            done1 = 1;
        }  /* switch choice 1 */
        switch(mode)
        {
        case 1:
            exp1 =  1.05;
            exp2 =  0.38;
            calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
         case 2:
            exp1 =  1.12;
            exp2 =  0.35;
                    calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
         case 3:
            exp1 =  1.20;
            exp2 =  0.32;
                    calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
    }
    /* Nominal productivity is one of 3 variables passed into */
    /* the simulation model that need algorithmic calculations. */
    /* This function will be discussed in detail below */
    fptr[23] = prod(PCNT,results[0],results[2],KDSI);
```

```
    }  /* while done1 */
}

/* This function gives the user the opportunity to view current */
/* values assigned to the COCOMO 15 cost drivers and make changes */
/* if necessary.  All cost drivers are defaulted to 1.00. */

void icocomo_in(float *rate,float *EAF1,int done3,char *fname1)
{
 /* rate is an array which hold the values for determining EAF for COCOMO */
    int choice,i;
    int CD1,CD2,CD3,CD4,CD5,CD6,CD7,CD8,CD9,CD10,CD11,CD12,CD13,CD14,CD15;

    /* clears screen and displays the 15 cost drivers and values */

    while(!done3)
    {
        clrscr();
   printf("           ********************************************\n");
   printf("               INTERMEDIATE LEVEL COCOMO MODEL INPUTS   \n");
   printf("                         for %s                \n",fname1);
   printf("           ********************************************\n\n");
   printf("                          1.  RELY:  %1.2f\n",rate[0]);
   printf("                          2.  DATA:  %1.2f\n",rate[1]);
   printf("                          3.  CPLX:  %1.2f\n",rate[2]);
   printf("                          4.  TIME:  %1.2f\n",rate[3]);
   printf("                          5.  STOR:  %1.2f\n",rate[4]);
   printf("                          6.  VIRT:  %1.2f\n",rate[5]);
   printf("                          7.  TURN:  %1.2f\n",rate[6]);
   printf("                          8.  ACAP:  %1.2f\n",rate[7]);
   printf("                          9.  AEXP:  %1.2f\n",rate[8]);
   printf("                         10.  PCAP:  %1.2f\n",rate[9]);
   printf("                         11.  VEXP:  %1.2f\n",rate[10]);
   printf("                         12.  LEXP:  %1.2f\n",rate[11]);
   printf("                         13.  MODP:  %1.2f\n",rate[12]);
   printf("                         14.  TOOL:  %1.2f\n",rate[13]);
   printf("                         15.  SCED:  %1.2f\n\n",rate[14]);
   printf("                              16.  Press [16 or 0] when entries are
complete.\n\n");

   /* allows user to select one of the above cost drivers by number */
   /* using the case statments the program calls specific pop-up */
   /* menus for the user to select specific values from and return */
   /* to display screen to see changes.  */

        printf("           Select Cost Driver and press [Enter]: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                CD1 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc1, TRUE);
                switch (CD1)
                {
                case 0:
                    break;

                case 1:
                    rate[0] = 0.75;
                    break;
                case 2:
                    rate[0] = 0.88;
```

```
        break;
    case 3:
        rate[0] = 1.00;
        break;
    case 4:
        rate[0] = 1.15;
        break;
    case 5:
        rate[0] = 1.40;
        break;
    }
    break;
case 2:
    CD2 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc2, TRUE);
    switch (CD2)
    {
    case 0:
        break;

    case 1:
        rate[1] = 0.94;
        break;
    case 2:
        rate[1] = 1.00;
        break;
    case 3:
        rate[1] = 1.08;
        break;
    case 4:
        rate[1] = 1.16;
        break;
    }
       break;
case 3:
    CD3 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc3, TRUE);
    switch (CD3)
    {
    case 0:
        break;

    case 1:
        rate[2] = 0.70;
        break;
    case 2:
        rate[2] = 0.85;
        break;
    case 3:
        rate[2] = 1.00;
        break;
    case 4:
        rate[2] = 1.15;
        break;
    case 5:
        rate[2] = 1.30;
        break;
    case 6:
        rate[2] = 1.65;
        break;
    }
            break;
case 4:
```

```
        CD4 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc4, TRUE);
        switch (CD4)
        {
        case 0:
            break;

        case 1:
            rate[3] = 1.00;
            break;
        case 2:
            rate[3] = 1.11;
            break;
        case 3:
            rate[3] = 1.30;
            break;
        case 4:
            rate[3] = 1.66;
            break;
        }
                break;
    case 5:
        CD5 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc5, TRUE);
        switch (CD5)
        {
        case 0:
            break;

        case 1:
            rate[4] = 1.00;
            break;
        case 2:
            rate[4] = 1.06;
            break;
        case 3:
            rate[4] = 1.21;
            break;
        case 4:
            rate[4] = 1.56;
            break;
        }
                break;
    case 6:
        CD6 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc6, TRUE);
        switch (CD6)
        {
        case 0:
            break;

        case 1:
            rate[5] = 0.87;
            break;
        case 2:
            rate[5] = 1.00;
            break;
        case 3:
            rate[5] = 1.15;
            break;
        case 4:
            rate[5] = 1.30;
            break;
        }
```

```
    break;
case 7:
    CD7 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc7, TRUE);
    switch (CD7)
    {
    case 0:
        break;

    case 1:
        rate[6] = 0.87;
        break;
    case 2:
        rate[6] = 1.00;
        break;
    case 3:
        rate[6] = 1.07;
        break;
    case 4:
        rate[6] = 1.15;
        break;
    }
    break;
case 8:
    CD8 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc8, TRUE);
    switch (CD8)
    {
    case 0:
        break;

    case 1:
        rate[7] = 1.46;
        break;
    case 2:
        rate[7] = 1.19;
        break;
    case 3:
        rate[7] = 1.00;
        break;
    case 4:
        rate[7] = 0.86;
        break;
            case 5:
        rate[7] = 0.71;
        break;
    }
            break;
case 9:
    CD9 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc9, TRUE);
    switch (CD9)
    {
    case 0:
        break;

    case 1:
        rate[8] = 1.29;
        break;
    case 2:
        rate[8] = 1.13;
        break;
    case 3:
        rate[8] = 1.00;
```

```
            break;
        case 4:
            rate[8] = 0.91;
            break;
        case 5:
            rate[8] = 0.82;
            break;
        }
                break;
    case 10:
        CD10 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc10, TRUE);
        switch (CD10)
        {
        case 0:
            break;

        case 1:
            rate[9] = 1.42;
            break;
        case 2:
            rate[9] = 1.17;
            break;
        case 3:
            rate[9] = 1.00;
            break;
        case 4:
            rate[9] = 0.86;
            break;
        case 5:
            rate[9] = 0.70;
            break;

        }
                break;
    case 11:
        CD11 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc11, TRUE);
        switch (CD11)
        {
        case 0:
            break;

        case 1:
            rate[10] = 1.21;
            break;
        case 2:
            rate[10] = 1.10;
            break;
        case 3:
            rate[10] = 1.00;
            break;
        case 4:
            rate[10] = 0.90;
            break;
        }
                break;
    case 12:
        CD12 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc12, TRUE);
        switch (CD12)
        {
        case 0:
            break;
```

```
    case 1:
        rate[11] = 1.14;
        break;
    case 2:
        rate[11] = 1.07;
        break;
    case 3:
        rate[11] = 1.00;
        break;
    case 4:
        rate[11] = 0.95;
        break;
    }
            break;
case 13:
    CD13 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc13, TRUE);
    switch (CD13)
    {
    case 0:
        break;

    case 1:
        rate[12] = 1.24;
        break;
    case 2:
        rate[12] = 1.10;
        break;
    case 3:
        rate[12] = 1.00;
        break;
    case 4:
        rate[12] = 0.91;
        break;
            case 5:
        rate[12] = 0.82;
        break;
    }
            break;
case 14:
    CD14 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc14, TRUE);
    switch (CD14)
    {
    case 0:
        break;

    case 1:
        rate[13] = 1.24;
        break;
    case 2:
        rate[13] = 1.10;
        break;
    case 3:
        rate[13] = 1.00;
        break;
    case 4:
        rate[13] = 0.91;
        break;
    case 5:
        rate[13] = 0.83;
        break;
    }
```

```
                    break;

            /* for the schedule cost driver need both the EAF value */
            /* but also the actual percent of schedule compression */
            /* or expansion for later COCOMO calculations. */

            case 15:
                CD15 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc15, TRUE);
                switch (CD15)
                {
                case 0:
                    break;

                case 1:
                    rate[14] = 1.23;
                    EAF1[0] = 0.75;
                    break;
                case 2:
                    rate[14] = 1.08;
                    EAF1[0] = 0.85;
                    break;
                case 3:
                    rate[14] = 1.00;
                    break;

                case 4:
                    rate[14] = 1.04;
                    EAF1[0] = 1.30;
                    break;
                case 5:
                    rate[14] = 1.10;
                    EAF1[0] = 1.60;
                    break;
                }
                        break;
            default:
                done3 = 1;
                break;
            }

    }    /* while done3 */
}

/* This function saves all of the current data for each project */
/* under a specific name specified by the user */

void file_save(float *DSMI,float *cdrate,float *PCNT,char *fname1,float EAF,int
*KDSI,int mode, float mf1)
{


    if ((fout = fopen(fname1,"wb"))==NULL)
    {
        fprintf(stderr,"Unable to open file %s \n",fname1);
    }
    else
     {
    printf("\n\n\nSaving ..............%s\n\n\n\n",fname1);
    printf("\n\n\nSaving ..............%s\n",fname1);
    fwrite((void *) DSMI,26 * sizeof(float),1,fout);
    fwrite((void *) KDSI,sizeof(int),1,fout);
```

```
   fwrite((void *) &EAF,sizeof(float),1,fout);
   fwrite((void *) cdrate,15 * sizeof(float),1,fout);
   fwrite((void *) PCNT,3 * sizeof(float),1,fout);
   fwrite((void *) &mode,sizeof(int),1,fout);
   fwrite((void *) &mf1,sizeof(float),1,fout);
   fclose(fout);
   }
}



/* This function provides the avenue to interface the output */
/* variables from this program into the simulation model via a */
/* text file and pass certain other variables to the TESTIO */
/* program via a binary file for reporting estimates and actual */
/* results and error rates. */


void file_prnt(float TOTMD1,float TDEV1,int *KDSI,float *DSMI)
{
   FILE *fpout;
   long l;

   /* need to change to long, once multiplied by 1000 */
   /* size could be out of integer range. */

   l = KDSI[0]*1000.0;

   /* Writes to textfile SIMONE.DNX */

   fprintf(textfile,"C RJBDSI=%ld\n", l);
   fprintf(textfile,"C TOTMD1=%5.2f\n", TOTMD1);
   fprintf(textfile,"C TDEV1=%5.2f\n", TDEV1);
   fprintf(textfile,"C INUDST=%5.2f\n", DSMI[0]);
   fprintf(textfile,"C ADMPPS=%5.2f\n", DSMI[1]);
   fprintf(textfile,"C HIREDY=%5.2f\n", DSMI[2]);
   fprintf(textfile,"C AVEMPT=%5.2f\n", DSMI[3]);
   fprintf(textfile,"C TRPNHR=%5.2f\n", DSMI[4]);
   fprintf(textfile,"C ASIMDY=%5.2f\n", DSMI[5]);
   fprintf(textfile,"T  TNERPK=%5.2f  %5.2f  %5.2f  %5.2f  %5.2f  %5.2f\n",
DSMI[6],DSMI[7],DSMI[8],DSMI[9],DSMI[10],DSMI[11]);
   fprintf(textfile,"T TPFMQA=%5.3f %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f
%   5   .   3   f          %   5   .   3   f          0   \   n   "   ,
DSMI[12],DSMI[13],DSMI[14],DSMI[15],DSMI[16],DSMI[17],DSMI[18],DSMI[19],DSMI[2
0],DSMI[21]);
   fprintf(textfile,"C DEVPRT=%5.2f\n", DSMI[22]);
   fprintf(textfile,"C DSIPTK=%5.2f\n", DSMI[23]);
   fclose(textfile);

   /* Writes to binary file OUTFILE.DNX */
   if ((fpout = fopen("outfile1.dnx","wb"))==NULL)
   {
        fprintf(stderr,"Unable to open file %s \n","outfile1.dnx");
   }
   else
   {

   /* write Estimated Effort and Estimated Schedule */
   fwrite((void *) &TOTMD1,sizeof(float),1,fpout);
   fwrite((void *) &TDEV1,sizeof(float),1,fpout);
   fclose(fpout);
```

```
    }
}

/* Part of the calculation for Nominal Productivity requires */
/* interpolation.  This function accepts staff size variable */
/* and returns communication overhead factor for use in determining */
/* Nominal Productivity. */

float interp(float stf_size)
{
    float covhd;

    if ((stf_size >= 0) && (stf_size <= 5))
    {
        covhd = (((stf_size-0)* .015)/5);
    }
    if ((stf_size > 5) && (stf_size <= 10))
    {
        covhd = (((stf_size-5)* .045)/5) +.015;
    }
    if ((stf_size > 10) && (stf_size <= 15))
    {
        covhd = (((stf_size-10)* .075)/5) + .06;
    }
    if ((stf_size > 15) && (stf_size <= 20))
    {
        covhd = (((stf_size-15)* .105)/5) + .135;
    }
    if ((stf_size > 20) && (stf_size <= 25))
    {
        covhd = (((stf_size-20)* .135)/5)  + .24;
    }
    if ((stf_size > 25) && (stf_size <= 30))
    {
        covhd = (((stf_size-25)* .165)/5) + .375;
    }
    if (stf_size >= 30)
    {
        covhd = .54;
    }
    return covhd;
}

/* This function does the Nominal Productivity calculations */
/* TOTMD1 - Effort passed from main function in man-days */
/* TDEV2  - Schedule in months not days! */
/* PCNT   - array from main function which passes %Testing,%QA */
/*          and %Rework for man-days */
/* MM     - Effort in man-months */
/* stf_size - Average Staff Size = MM/TDEV2 */
/* DEVMD - Development man-days */
/* ADP    - Actual Development Productivity */
/* covhd - Communication Overhead */
/* product - Nominal Productivity */

float prod(float *PCNT,float TOTMD1,float TDEV2,int *KDSI)
{
        float MM,stf_size,DEVMD,ADP,covhd;
        float product;

        MM = TOTMD1/19;
```

```
        stf_size = MM/TDEV2;
        DEVMD = (1-(PCNT[0]+PCNT[1]+PCNT[2]))*TOTMD1;
        ADP = (KDSI[0] * 1000.0)/DEVMD;
        covhd = interp(stf_size);    /* call interpolation function */
        product = ADP/(0.6 * (1.0-covhd));
        return product;

}

/* This function completes COCOMO calculations for input into */
/* simulation model */
/* result array is used to hold man-day and schedule results */
/* EAF1 contains the percent to multiply TDEV by from cost driver 15 */
/* mf1, exp1 and exp2 are the coefficients and exponents passed */
/* in from main function */

void calc(float *result,int *KDSI,float *EAF1,float *cdrate,float mf1,float
exp1,float exp2)
{
   int i;
   float EAF;      /* Estimated Adjustment Factor */
   EAF = 1.00;
   for (i=0;i<15;i++)
   {
        EAF *= cdrate[i];    /* Calculate the EAF by multiplying each */
                             /* cost driver by one another */
   }
   /* Total man-days calculation */
   result[0] =  mf1 * (pow(KDSI[0],exp1)) * 19.0 * EAF;

   /* if cost driver 15 (schedule) is nominal then calculations */
   /* are straight forward. If not you must divide the man-days */
   /* by cdrate[14] or calculate total man-days as if schedule */
   /* was nominal. */

   if (EAF1[0] != 1.00)
   {
        result[1] =  2.5 * pow(((result[0]/19.0)/cdrate[14]),exp2) * EAF1[0] *
19.0;
        result[2] =  2.5 * pow(((result[0]/cdrate[14])/19.0),exp2) * EAF1[0];
   }
   else
   {
        result[1] =  2.5 * pow((result[0]/19.0),exp2) * 19.0;
        result[2] =  2.5 * pow((result[0]/19.0),exp2);
   }
   result[3] = EAF;
   return;
}

/* Small function that simply initializes all the */
/* cost drivers to 1 */

void initial(float *cdrate)
{
   int i;
   for(i=0;i < 15; i++)
   {
        cdrate[i] = 1.00;
   }
    return;
```

```
}


void main()
{
  int i, done=0,done1=0, done3=0, done5=0, done6=0;
  int sel, sel1, sel2, sel3, sel4, sel5;
  int mode;               /* One of 3 COCOMO modes */
  int KDSI[2];            /* Stores size and counter */
  float EAF1[1];          /* Stores schedule cost driver percent */
  float cdrate[15], DSMI[26], PCNT[3],results[4];
  float TOTMD1, TDEV1, TDEV2, ADP, mf1, exp1, exp2;
  char fname1[13];
  char fname2[13];
  char string[25];
  char string1[25];
  int ch, basic;

  /* creates textfile which is interface with simulation model */
  textfile = fopen("SIMONE.DNX","w");

  /* initializes scedule cost driver to 1 */
  EAF1[0] = 1.00;
  /* bat is the boarder attribute for the pop-up window */
  /* sets background to blue and boarder to white */
  bat = v_setatr(BLUE,WHITE,0,0);

  /* wat is the window attribute for the pop-up window */
  /* sets background to blue and text to white */
  wat = v_setatr(BLUE,WHITE,0,0);

  clrscr();
 /* this while statement gets program started always initiated on */
 while (!done6)
 {
   clrscr();            /* pop-up initial menu */
                         /* allows user to go to main menu */
                         /* create a new project or EXIT */
   sel = wn_popup(0, 5, 10, 55, 10, wat, bat, &intelc, TRUE);

   switch (sel)
   {
   case 1:     /* user selected to go to main menu */
    done5=0;
    while(!done5)
    {
      clrscr();                /* Main Menu will allow user to */
                        /* list, select, run simulation */
                        /* or exit this menu    */
      sel2 = wn_popup(0, 5, 10, 55, 12, wat, bat, &intelc0, TRUE);

      switch (sel2)
      {
         case 1:      /* user selected to list data files */
             filelist();   /* calls function to list all *.prf files */
                           /* no break; continues to case 2 */
         case 2:           /* user can look at list and enter file name */
             printf("\n\n:>  Enter project filename: ");
             scanf("%s",&fname1);
             /* if name of file is mis-entered program goes back to */
             /* main menu */
```

```
if ((fin = fopen(fname1,"rb"))==NULL)
{
   fprintf(stderr,"Unable to open file %s to read\n",fname1);
   continue;
}
/* read in the data of the selected filename */
fread((void *) DSMI,26 * sizeof(float),1,fin);
fread((void *) KDSI,sizeof(int),1,fin);
fread((void *) &results[3],sizeof(float),1,fin);
fread((void *) cdrate,15 * sizeof(float),1,fin);
fread((void *) PCNT,3 * sizeof(float),1,fin);
fread((void *) &mode,sizeof(int),1,fin);
fread((void *) &mf1,sizeof(float),1,fin);
fclose(fin);
/* re-intialize */
done1 = 0;

/* function described above which allows user to display */
/* on screen the variables for the simulation model less */
/* the COCOMO variables */

model_in(DSMI,PCNT,KDSI,results,mode,fname1,EAF1,cdrate,done1,mf1);

if(results[3]==1.00)  /* checks if EAF = 1.00 */
{
   clrscr();          /* gives user option to continue */
                /* using basic model or use intermediate */
   sel3 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc21, TRUE);
   switch(sel3)
   {
       case 1:      /* user selected basic model */
           basic = 1;
           initial(cdrate);  /* Basic model EAF values must */
           break;            /* all be 1.00.  This sets all */
           /* cost drivers to 1.00 */
       case 2:  /* user selected intermediate model */
           basic = 0;
           initial(cdrate);
           /* Displays cost driver screen; allows user to */
           /* set cost drivers to desired level */
           icocomo_in(cdrate,EAF1,done3,fname1);
           break;
   }  /* switch sel3 */
}  /* if */
else  /* if EAF is other than 1.00 */
{
   /* displays cost drivers values and */
   /* allows user to manipulate */
   basic = 0;
   icocomo_in(cdrate,EAF1,done3,fname1);
}
clrscr();
   /* pop-up menu for user to select COCOMO mode */
mode = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc20, TRUE);

switch (mode)
{
   case 1:             /* Organic mode */
       if (basic != 1)
       {
           mf1  =  3.2;    /* sets coefficient and exponents */
```

```
            }
            else
            {
                mf1  =  2.4;
            }
            exp1 =  1.05;   /* for man-days calculation */
            exp2 =  0.38;   /* for schedule calculation */
            /* function that actually does the COCOMO calculations */
            calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
        case 2:             /* Semi-detached mode */
                mf1  =  3.0;
            exp1 =  1.12;
            exp2 =  0.35;
                calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
        case 3:                 /* Embedded mode */
            if (basic != 1)
            {
                mf1  =  3.6;    /* sets coefficient and exponents */
            }
            else
            {
                mf1  =  2.8;
            }
            exp1 =  1.20;
            exp2 =  0.32;
                calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
     } /* switch mode */

    /* allows user to save current datafile under the same name */
     /* or save the same or manipulated data under a new name */
     sel4 = wn_popup(0, 5, 10, 55, 10, wat, bat, &intelc22, TRUE);

     switch (sel4)
     {
        case 1:  /* save in same file */
            file_save(DSMI,cdrate,PCNT,fname1,
                results[3],KDSI,mode,mf1);
            break;
        case 2:     /* if you are changing the name of the file */
            clrscr();
            gotoxy(12,10);      /* enter new name */
            printf("Enter the new project filename:  ");
            scanf("%s",&string);

        /* Program lets user enter more than 8 characters */
        /* for filename.  This copies first 8 characters */
        /* into nem filename variable */
            strncpy(fname2, string, 8);
            string[8] = '\0';    /* resets string to null */
            strcat(fname2,".prf"); /* automatically adds ".prf" */

                  /* writes new file to disk */
            fnew = fopen(fname2,"wb");
            fwrite((void *) DSMI,26 * sizeof(float),1,fnew);
            fwrite((void *) KDSI,sizeof(int),1,fnew);
            fwrite((void *) &results[3],sizeof(float), 1,fnew);
            fwrite((void *) cdrate,15 * sizeof(float),1,fnew);
            fwrite((void *) PCNT,3 * sizeof(float),1,fnew);
```

```
                    fwrite((void *) &mode,sizeof(int),1,fnew);
                        fwrite((void *) &mf1,sizeof(float),1,fnew);
                    fclose(fnew);
                }
                break;
            case 3:      /* allows user to exit into the simulation */
                         /* model automatically saving data first */
                done5 = 1;  /* Will cause exit to main menu */
                done6 = 1;  /* Will cause exit from program */

                /* calls file_prnt which outputs both SIMONE.DNX */
                /* and OUTFILE.DNX.  After completion of this function */
                /* exit program to DOS which calls Simulation Model */
                file_prnt(results[0],results[1],KDSI,DSMI);
                break;

            case 4:
                done5 = 1;      /* Exits to main menu only */
                done6 = 0;
                break;
            }
        }
        break;

    case 2:          /* your selection was to create a new project */

        wn_init();  /* initialize a window for text entry */
        w4 = wn_open(0,5,10,58,12,wat,bat); /* open window w4; similar */
                        /* to opening a file */
         /* 5 is starting row; 10 is starting column; 58 is characters wide */
        /* second 12 is number of rows; wat is the window attribute and */
        /* bat the boarder attribute */
        if(!w4) exit(1);
        wn_printf(w4,"        \n    ********************    IMPORTANT
                        ********************\n");
        wn_printf(w4,"  \n\n  In order to load a NEW project you must enter\n");
        wn_printf(w4,"     input  data  for  both  COCOMO  and  the  Dynamic
                        Simulation.\n");
        wn_printf(w4,"    There  are  two  forms  on  which  all  data  must  be
                        entered.\n");
        wn_printf(w4,"       Please  enter  the  data  as  accurately  as
                        possible.\n\n\n\n");
        wn_printf(w4,"            Press [ANY KEY] to continue...");

    /* wn_printf works similar to fprintf: prints to window vice file */
        v_getch();
        wn_close(w4);

        /* Front end; allows user to make necessary inputs for the */
        /* Dynamic Simulation Model; Inputs appear one at a time and */
        /* there must be an entry for each variable; all inputs will */
        /* be stored in an array DSMI */
        clrscr();
printf("                 ****************************************\n");
printf("                *   DYNAMIC SIMULATION MODEL INPUTS    *\n");
printf("                ****************************************\n\n");
printf("          Input the following: \n\n");
printf("                1.  Initial Under Staffing Level Factor:  ");
        scanf("%f",&DSMI[0]);
printf("                2.  Average Daily Manpower per Staff Member:  ");
        scanf("%f",&DSMI[1]);
```

```
printf("                3.  Hiring Delay:  ");
        scanf("%f",&DSMI[2]);
printf("                4.  Average Employment Time:  ");
        scanf("%f",&DSMI[3]);
printf("                5.  Training Overhead:  ");
        scanf("%f",&DSMI[4]);
printf("                6.  Average Assimilation Delay:  ");
        scanf("%f",&DSMI[5]);
printf("                7.  Error Rate (Must enter 6 input values): \n");
        for(i=0; i < 6; i++)
        {
printf("                          Error rate[%d]: ",(i+1));
            scanf("%f",&DSMI[i+6]);
        }
printf("                8.  Planned Fraction of Manpower for QA \n");
printf("                    (Must enter 10 input values):  \n");
        for(i=0; i < 10; i++)
        {
printf("                           Manpower for QA[%d]: ",(i+1));
            scanf("%f",&DSMI[i+12]);
        }
printf("                9.  DEVPRT:  ");
        scanf("%f",&DSMI[22]);
        /* entries to the PCNT array are for nominal productivity calculation */
printf("                  10.  Nominal Potential Productivity Man Day percent
                        inputs:  \n\n");
printf("                                Percent MD for tests:  ");
        scanf("%f",&PCNT[0]);
printf("\n              (Please be consistent with TPFMQA input values)\n");
printf("                                Percent MD for QA:     ");
        scanf("%f",&PCNT[1]);
printf("                                Percent MD for Rework: ");
        scanf("%f",&PCNT[2]);

        /* Allows user to make necessary inputs for the */
        /* COCOMO Model; Inputs appear one at a time and */
        /* there must be an entry for each variable */
        clrscr();
printf("                ***************************************\n");
printf("                *          COCOMO MODEL INPUTS          *\n");
printf("                ***************************************\n\n\n");
printf("           Input the following: \n\n");
printf("                1.  Estimated Project Size in KDSI:  ");
        scanf("%d",&KDSI[0]);  /* string gets KDSI value  */

printf("                2.  Enter the Project Name: ");
        scanf("%s",&string1);   /* string gets project name */

        strncpy(fname1, string1, 8); /* since dos only recognizes the first */
                    /* 8 characters fname1 takes first 8 */
                    /* characters in string1 */
        string[8] = '\0';    /* resets string1 to null set so next */
                    /* project name if short will not contain */
                    /* characters previously resident in string1 */
        strcat(fname1,".prf"); /* automatically tags all project names */
                      /* with .prf to easily recognize projects */

        clrscr();
        sel1 = wn_popup(0, 5, 15, 55, 10, wat, bat, &intelc19, TRUE);
                  /* selection of basic or intermediate COCOMO */
        switch(sel1)
```

```
{
case 1:    /* user selection basic */
    basic = 1;
    EAF1[0] = 1.00;    /* initializes schedule percent to 100 */
    initial(cdrate);   /* sets all cost drivers to nominal */
    break;
case 2:    /* user selection intermediate */
    basic = 0;
    done3 = 0 ;
    EAF1[0] = 1.00;    /* initializes schedule percent to 100 */
    initial(cdrate);   /* sets all cost drivers to nominal */
    icocomo_in(cdrate,EAF1,done3,fname1); /* allows user to set */
                    /* cost driver values */
    break;
}
clrscr();

mode = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc20, TRUE);
                  /* select a mode */
switch (mode)
{
    case 1:      /* Organic */
        if (basic != 1)
        {
            mf1  =  3.2;    /* sets coefficient and exponents */
        }
        else
        {
            mf1  =  2.4;
        }
        exp1 =  1.05;
        exp2 =  0.38;
        /* calls function to do COCOMO calculations */
                calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
     case 2:      /* Semi-detached */
        mf1  =  3.0;
        exp1 =  1.12;
        exp2 =  0.35;
        /* calls function to do COCOMO calculations */
                calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
     case 3:      /* Embedded */
        if (basic != 1)
        {
            mf1  =  3.6;    /* sets coefficient and exponents */
        }

        else
        {
            mf1  =  2.8;
        }
        exp1 =  1.20;
        exp2 =  0.32;
        /* calls function to do COCOMO calculations */
                calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
}
/* calls function to do nominal productivity calculations */
/* results[0]=Total man-days; results[2]=schedule in months */
DSMI[23] = prod(PCNT,results[0],results[2],KDSI);
```

```
        /* display/edit simulation model inputs */
        model_in(DSMI,PCNT,KDSI,results,mode,fname1,EAF1,cdrate,done1,mf1);

        /* save updated file automatically */
        file_save(DSMI,cdrate,PCNT,fname1,results[3],KDSI,mode,mf1);

        /* this ends the input phase and initial COCOMO calculations */
        /* the user now goes to NEW PROJECT MENU which enables the user */
        /* to Display/Edit or run simulation model */
        done = 0;
        while(!done)    /* New Project Menu loop */
        {
            clrscr();
            sel5 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc23, TRUE);
                   /* New Project Menu */
            switch (sel5)
            {
            case 1:            /* user selected Display/Edit */
                /* display or edit simulation model inputs */
                           model_in(DSMI,PCNT,KDSI,results,mode,
                      fname1,EAF1,cdrate,done1,mf1);

                if(results[3]==1.00) /* checks if EAF = 1.00 */
                {
                    clrscr();  /* gives user option to continue */
                    /* using basic model or use intermediate */
                    sel3 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc21,
TRUE);
                    switch(sel3)
                    {
                        case 1:  /* user selected basic model */
                        basic = 1;
                        initial(cdrate); /* Basic model EAF values must*/
                        break;           /* all be 1.00.  This sets all */
                         /* cost drivers to 1.00 */
                        case 2:        /* user selected intermediate model */
                        initial(cdrate);
                           /* Displays cost driver screen; allows user to */
                     /* set cost drivers to desired level */
                        basic = 0;
                        icocomo_in(cdrate,EAF1,
                        done3,fname1);
                        break;
                    }  /* switch sel3 */
                }  /* if */
                else
                /* if EAF is other than 1.00 */
                {
                    /* displays cost drivers values and */
                    /* allows user to manipulate */
                    basic = 0;
                    icocomo_in(cdrate,EAF1,done3,fname1);
                }
                clrscr();
                switch (mode)
                {
                    case 1:   /* Organic mode */
                        if (basic != 1)
                        {
                        mf1  =  3.2;    /* sets coefficient and exponents */
                        }
```

```
                    else
                    {
                    mf1  =  2.4;
                    }
                    exp1 =  1.05; /* for man-days calculation */
                    exp2 =  0.38;
                /* function that actually does the COCOMO calculations */
                   calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
                    break;
                case 2:    /* Semi-detached mode */
                    mf1  =  3.0;
                    exp1 =  1.12;
                    exp2 =  0.35;
                    calc(results,KDSI,EAF1,cdrate,mf1,
                    exp1,exp2);
                    break;
                case 3:     /* Embedded mode */
                    if (basic != 1)
                    {
                    mf1  =  3.6;    /* sets coefficient and exponents */
                    }
                    else
                    {
                    mf1  =  2.8;
                    }
                    exp1 =  1.20;
                    exp2 =  0.32;
                   calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
                    break;
            } /* switch mode */
            /* automatically save latest changes to file */
            file_save(DSMI,cdrate,PCNT,fname1,results[3],KDSI,mode,mf1);
            break;
        case 2:    /* user selected to run simulation model */
            done = 1;    /* allows user to exit to initial menu */
            done6 = 1;   /* allows user to exit out of program */

        /* calls file_prnt which outputs both SIMONE.DNX */
          /* and OUTFILE.DNX.  After completion of this function */
        /* exit program to DOS which calls Simulation Model */
            file_prnt(results[0],results[1],KDSI,DSMI);
            break;
        case 3:
            done = 1;  /* allows user to exit to initial menu */
            break;
        }
    }
    break;
  case 3:
    exit(1);  /* EXITS FROM PROGRAM WITHOUT DOING SIMULATION */
 }


}
exit(0);  /* EXITS FROM PROGRAM WITH DOING SIMULATION */
}
```

APPENDIX C

```
/* ********************************************************** */
/* * Author: Richard W. Smith    Advisor:  Prof. Abdel-Hamid  * */
/* * Program: OUTPUT1     Lang: C                           * */
/* * Used Shareware <windows.h> in project environment      * */
/* ********************************************************** */

/* This is one of 5 programs written and interfaced with the */
/* Dynamic Simulation Model.  This particular program gathers  */
/* outfile information from several files to generate reports. */

/* The following headers were used and needed to utilize the */
/* library functions used throughout this program. */

#include <stdio.h>
#include <math.h>
#include <conio.h>


void main(void)
{
   /* Declarations for variables used within this program */

   int i, k=0, m=0, KDSI[2], count, num;
   float j, md, time, TOTMD1, TDEV1, diff, diff1;
   char string[8], dest[8], string1[8], dest1[8];
   FILE *fpin, *fdata, *results;

    /* initializes the dest(destination) string as null */
   for (i=0;i<8;i++)
        dest[i] = '\0';

   /* Read the outfile.dnx:  Binary file used in reporting */
   /* Easier to work with binary in this case */

   if ((fdata = fopen("outfile1.dnx","rb"))==NULL)
   {
        fprintf(stderr,"Unable to open file %s \n","outfile1.dnx");
   }
   else                    /* reads effort and schedule */
   {
   fread((void *) &TOTMD1,sizeof(float),1,fdata);
    fread((void *) &TDEV1,sizeof(float),1,fdata);
   fclose(fdata);
   }

   fpin = fopen("SIMONE.OUT","r");  /* get output from simulation */

            /* GET EFFORT VALUE FROM SIMULATION OUTPUT FILE */
   i = fgetc(fpin);      /* get first character from output file */

   while (i != 40)       /* continue getting characters until ascii */
   {                     /* #40 '(' */
        i = fgetc(fpin);
   }
```

```
i = fgetc(fpin);
while (i != 41)       /* now get each char and save as string */
{
     string[k] = i;
     k++;
     i = fgetc(fpin);
}
string[k] = '\0';
i = fgetc(fpin);

           /* GET SCHEDULE VALUE FROM SIMULATION OUTPUT FILE */
while (i != 40)
{
     i = fgetc(fpin);
}
i = fgetc(fpin);
while (i != 41)         /* continue getting characters until ascii */
{                       /* #41 ')' */
     string1[m] = i;
     m++;
     i = fgetc(fpin);
}
string1[m] = '\0';
fclose(fpin);
strncpy(dest,string,k);         /* copy actual effort into dest string */
strncpy(dest1,string1,m);       /* copy actual sked into dest1 string */
md = atof(dest);                /* string to float conversion */
time = atof(dest1);

if (TOTMD1 >= md)          /* checks if estimated effort is > actual */
{
     if (TDEV1 >= time)    /* checks if estimated sked is > actual */
     {
         diff = (TOTMD1-md)/md;      /* calc error rates for effort */
         diff1 = (TDEV1-time)/time; /* calc error rates for sked */
     }
     else
     {
         diff = (TOTMD1-md)/md;      /* calc error rates for effort */
         diff1 = (time-TDEV1)/time; /* calc error rates for sked */
     }
}
else
{
     if (TDEV1 >= time)  /* checks if estimated sked is > actual */
     {
         diff = (md-TOTMD1)/md;     /* calc error rates for effort */
         diff1 = (TDEV1-time)/time; /* calc error rates for sked */
     }
     else
     {
         diff = (md-TOTMD1)/md;     /* calc error rates for effort */
         diff1 = (time-TDEV1)/time;/* calc error rates for sked */
     }
}

/* open report file */
results = fopen("REPORT.OUT","a");

/* output format */
```

```
   fprintf(results,"\n\n\nEstimated   Actual     Percent   Estimated   Actual
  Percent\n");
   fprintf(results,"man-days    man-days   Error     Schedule    Schedule   Er
\n");
   fprintf(results," %6.0f      %6.0f  %6.0f      %6.0f       %6.0f     %6.0f
           \n",TOTMD1,md,diff,TDEV1,time,diff1);
   fprintf(results,"\n\n**** This data is available in REPORT.OUT ****\n");
   fprintf(results,"**** Each time the model is run REPORT.OUT will change
                    ****\n");
   fclose(results);

   /* display for printing report or displaying the report on screen */
   gotoxy(15,10);
   printf("      REPORT FORMAT CHOICE\n");
   gotoxy(15,12);
   printf("       1 - Display results\n");
   gotoxy(15,13);
   printf("       2 - Print results\n");
   gotoxy(15,14);
   printf("       3 - Exit\n");
   gotoxy(15,16);
   printf("Enter one of the above:  ");
   scanf("%d",&num);

   switch(num)         /* case statement exits program or to DOS */
   {
        case 1:
            clrscr();
            exit(4);   /* exit to DOS and display results on screen */
        case 2:
            clrscr();
            exit(3);   /* exit to DOS and send results to printer */
        case 3:
            exit(0);   /* exit program */
   }
}

/* end output1 program */
```

APPENDIX D

```
/* ********************************************************** */
/* * Author: Richard W. Smith    Advisor:  Prof. Abdel-Hamid  * */
/* * Program: Input2             Lang: C                      * */
/* * Used Shareware <windows.h> in project environment        * */
/* ********************************************************** */

/* This is one of 5 programs written and interfaced with the */
/* Dynamic Simulation Model.  This particular program completes */
/* two tasks.  First it accepts input variables for the dynamic */
/* simulation model and COCOMO acting as a front end for the  */
/* model in the two project environment.  Then it makes all */
/* the necessary COCOMO calcuations for either the Basic or  */
/* the intermediate versions of COCOMO for each project. */


/* The following headers were used and needed to utilize the */
/* library functions used throughout this program. */

#include <windows.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <dir.h>
#include <string.h>
/* Prototypes for the functions which will be */
/* described below.                           */

int filelist(void);
void model_in(float *,float *,int *,float *,int,char a[],float *,
                    float*,int,float);
void icocomo_in(float *,float *,int,char *);
void file_save(float *,float *,float *,char *,float,int *,int,float);
void file_pass(float,float,float,float,int *,int *,float *,float *,
                    float *,float *);
float interp(float);
float prod(float *,float,float,int *);
void calc(float *,int *,float *,float *,float,float,float);
void initial(float *);
void dsm_in(float *, float *);
void como_in(int *,char *);

/* Declarations: */
int bat;                        /* border atrib */
int wat;                        /* window atrib */

/*Pointer to files being used*/
FILE * textfile;
FILE * fin, *fin2;
FILE * fout, *fout2;
FILE * fnew, *fnew2;
WINDOWPTR w3;                   /* window declaration */
WINDOWPTR w4;                   /* window declaration */
```

```
/* The following are static structures developed to be */
/* used throughout the program in pop-up menus for various */
/* user selection requirements.  The learning curve for */
/* the use of windows.h was considerable, however, once */
/* learned it is fairly simple to create menus. */



static struct pmenu intelc =
 {0, FALSE, 0,          /* Must be FALSE */
 1, 3,                  /* The 1 initiates which row */
                    /* The 3 determines number of lines */
                    /* The 3 determines number of lines */
                    /* which can be highlighted after row */
 /* row,col */
 1, 20, "INITIAL MENU", 0,
 4, 12, "1 - LOAD projects from disk.", 1,
 5, 12, "2 - NEW projects.", 2,
 6, 12, "3 - EXIT Program.", 3,
 9, 3, "Select with number or cursor and press [ENTER]...",0,
99, 99, "",99
 };

static struct pmenu intelc25 =
 {0, FALSE, 0,
 1, 3,
 1, 12, "  SELECT NEW PROJECT MENU", 0,
 4, 12, "1 - ENTER New Project 1.", 1,
 5, 12, "2 - ENTER New Project 2.", 2,
 6, 12, "3 - EDIT/DISPLAY/RUN.", 3,
 9, 3, "Select with number or cursor and press [ENTER]...",0,
99, 99, "",99
 };


static struct pmenu intelc23 =
 {0, FALSE, 0,
 1, 4,
 1, 20, "NEW PROJECT MENU", 0,
 4, 12, "1 - Display/Edit Project 1.", 1,
 5, 12, "2 - Display/Edit Project 2.", 2,
 6, 12, "3 - RUN Dynamic Simulation.", 3,
 7, 12, "4 - QUIT menu.", 4,
 9, 3, "Select with number or cursor and press [ENTER]...",0,
99, 99, "",99
 };

static struct pmenu intelc0 =
 {0, FALSE, 0,
 1, 4,
 1, 21, "   MAIN MENU", 0,
 3, 15, "1 - SELECT Project 1 from disk.", 1,
 4, 15, "2 - SELECT Project 2 from disk.", 2,
 5, 15, "3 - RUN Dynamic Simulation.",3,
 6, 15, "4 - QUIT menu.", 4,
 9, 3, "Select with number or cursor and press [ENTER]...",0,
99, 99, "",99
 };
```

```
static struct pmenu intelc19 =
 {0, FALSE, 0,
 2, 3,
 1, 15, "       COCOMO MODEL", 0,
 2, 15, "        ESC - EXIT ", 0,
 4, 15, "1 - Basic COCOMO Model", 1,
 5, 15, "2 - Intermediate COCOMO Model", 2,
 7, 3, "Select the model you wish to use and press enter:",0,
99, 99, "",99
 };


static struct pmenu intelc21 =
 {0, FALSE, 0,
 2, 3,
 1, 6, "Current data file is for the Basic COCOMO", 0,
 2, 6, "    (Select one of the following)", 0,
 4, 10, "1 - CONTINUE Basic COCOMO Model", 1,
 5, 10, "2 - Intermediate COCOMO Model", 2,
 7, 3, "Select the model you wish to use and press enter:",0,
99, 99, "",99
 };

static struct pmenu intelc22 =
 {0, FALSE, 0,
 2, 3,
 1, 16, "SAVING FILES", 0,
 2, 6, "    (Select one of the following)", 0,
 4, 10, "1 - SAVE changes under Same Name ", 1,
 5, 10, "2 - SAVE changes under New Name", 2,
 7, 3, "Select the model you wish to use and press enter:",0,
99, 99, "",99
 };

static struct pmenu intelc20 =
 {0, FALSE, 0,
 2, 4,
 1, 10, "      COCOMO MODE SELECTION", 0,
 2, 10, "           ESC - EXIT ", 0,
 4, 18, "1 - Organic", 1,
 5, 18, "2 - Semi-detached", 2,
 6, 18, "3 - Embedded",3,
 8, 3, "Select the appropriate mode and press enter:  ",0,
99, 99, "",99
 };

static struct pmenu intelc1 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "       RELY(Required software reliability)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 0.75", 1,
 5, 15, "2 - Low; 0.88", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 1.15",4,
 8, 15, "5 - Very High; 1.40", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc2 =
```

```
 {0, FALSE, 0,
 2, 5,
 1, 2, "            DATA(Database size):", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Low; 0.94", 1,
 5, 15, "2 - Nominal; 1.00", 2,
 6, 15, "3 - High; 1.08", 3,
 7, 15, "4 - Very High; 1.16",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };



static struct pmenu intelc3 =
 {0, FALSE, 0,
 2, 7,
 1, 2, "           CPLX(Product complexity)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1-Very Low; 0.70",1,
 5, 15, "2-Low; 0.85",2,
 6, 15, "3-Nominal; 1.00", 3,
 7, 15, "4-High; 1.15",4,
 8, 15, "5-Very High; 1.30", 5,
 9, 15, "6-Extra High; 1.65", 6,
 11, 1, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc4 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "         TIME(Exection time constraint)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Nominal; 1.00", 1,
 5, 15, "2 - High; 1.11", 2,
 6, 15, "3 - Very High; 1.30", 3,
 7, 15, "4 - Extra High; 1.66",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc5 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "         STOR(Main storage constraint)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Nominal; 1.00", 1,
 5, 15, "2 - High; 1.06", 2,
 6, 15, "3 - Very High; 1.21", 3,
 7, 15, "4 - Extra High; 1.56",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc6 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "       VIRT(Virtual machine volatility)", 0,
 2, 2, "                ESC - EXIT ", 0,
```

```
 4, 15, "1 - Low; 0.87", 1,
 5, 15, "2 - Nominal; 1.00", 2,
 6, 15, "3 - High; 1.15", 3,
 7, 15, "4 - Very High; 1.30", 4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc7 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "        TURN(Computer turnaround time)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Low; 0.87", 1,
 5, 15, "2 - Nominal; 1.00", 2,
 6, 15, "3 - High; 1.07", 3,
 7, 15, "4 - Very High; 1.15",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc8 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "           ACAP(Analyst capability)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.46", 1,
 5, 15, "2 - Low; 1.19", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.86",4,
 8, 15, "5 - Very High; 0.71", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc9 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "         AEXP(Applications experience)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.29", 1,
 5, 15, "2 - Low; 1.13", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.91",4,
 8, 15, "5 - Very High; 0.82", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc10 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "         PCAP(Programmer capability)", 0,
 2, 2, "                ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.42", 1,
 5, 15, "2 - Low; 1.17", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.86",4,
 8, 15, "5 - Very High; 0.70", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
```

```
 };

static struct pmenu intelc11 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "       VEXP(Virtual machine experience)", 0,
 2, 2, "                  ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.21", 1,
 5, 15, "2 - Low; 1.10", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.90",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc12 =
 {0, FALSE, 0,
 2, 5,
 1, 2, "      LEXP(Programming Language experience)", 0,
 2, 2, "                   ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.14", 1,
 5, 15, "2 - Low; 1.07", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.95",4,
10, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc13 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "       MODP(Use of modern programming practices)", 0,
 2, 2, "                     ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.24", 1,
 5, 15, "2 - Low; 1.10", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.91", 4,
 8, 15, "5 - Very High; 0.82", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc14 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "           TOOL(Use of software tools)", 0,
 2, 2, "                   ESC - EXIT ", 0,
 4, 15, "1 - Very Low; 1.24", 1,
 5, 15, "2 - Low; 1.10", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 0.91", 4,
 8, 15, "5 - Very High; 0.83", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
 };

static struct pmenu intelc15 =
 {0, FALSE, 0,
 2, 6,
 1, 2, "      SCED(Required development schedule", 0,
 2, 2, "                  ESC - EXIT ", 0,
```

```
 4, 15, "1 - Very Low; 1.23", 1,
 5, 15, "2 - Low; 1.08", 2,
 6, 15, "3 - Nominal; 1.00", 3,
 7, 15, "4 - High; 1.04",4,
 8, 15, "5 - Very High; 1.10", 5,
11, 3, "Select the appropriate Software Cost Driver Rating: ",0,
99, 99, "",99
};

/* Function which lists all the data files (*.prf) */
/* in the current directory.  */

int filelist(void)
{
   struct ffblk ffblk;
   int done4;
   printf("Data file listing: \n\n");
   done4 = findfirst("*.prf",&ffblk,0); /* finds first .prf file */
   while(!done4)
   {
        printf("   %s\n",ffblk.ff_name);
        done4 = findnext(&ffblk);        /* finds the next .prf file */
   }
   return(1);
}

/* This function accepts numerous pointers to various strings */
/* which allows the user to select variables from the display */
/* and change the value of current simulation input variables. */

void model_in(float *fptr,float *PCNT,int *KDSI,float *results,int mode,char
fname1[],float *EAF1,float *cdrate,int done1,float mf1)
{
   /* Declarations for this function */
   int choice1, choice2, choice3;
   float exp1,exp2;
   char string1[] = "Organic";
   char string2[] = "Semi-detached";
   char string3[] = "Embedded";
   char string[14];

   switch(mode)    /* mode variable is passed in to function */
   {               /* above for display on this screen; switch/case format*/
                   /* above for display on this screen; switch/case format*/
   case 1:
        strcpy(string,string1);
        break;
   case 2:
        strcpy(string,string2);
        break;
   case 3:
        strcpy(string,string3);
        break;
   }

   /* clears screen and displays variables on screen in below format */

   while(!done1)
   {
        clrscr();
   printf("                ****************************************\n");
```

```
printf("                    MODEL INPUTS for %s       \n",fname1);
  printf("                      %s Mode\n",string);
printf("                 ****************************************\n\n");
printf("                  1. INUDST:  %5.3f                8.  (1) TPFMQA[1]:
                %5.3f\n",fptr[0],fptr[12]);
printf("                  2. ADMPPS:  %5.3f                    (2) TPFMQA[2]:
                %5.3f\n",fptr[1],fptr[13]);
printf("                  3. HIREDY:  %5.3f                    (3) TPFMQA[3]:
                %5.3f\n",fptr[2],fptr[14]);
printf("                   4. AVEMPT:  %5.3f                   (4) TPFMQA[4]:
                %5.3f\n",fptr[3],fptr[15]);
printf("                  5. TRPNHR:  %5.3f                    (5) TPFMQA[5]:
                %5.3f\n",fptr[4],fptr[16]);
printf("                  6. ASIMDY:  %5.3f                    (6) TPFMQA[6]:
                %5.3f\n",fptr[5],fptr[17]);
printf("                   7. (1) TNERPK[1]:  %5.3f            (7) TPFMQA[7]:
                %5.3f\n",fptr[6],fptr[18]);
printf("                         (2) TNERPK[2]:  %5.3f         (8) TPFMQA[8]:
                %5.3f\n",fptr[7],fptr[19]);
printf("                         (3) TNERPK[3]:  %5.3f         (9) TPFMQA[9]:
                %5.3f\n",fptr[8],fptr[20]);
printf("                        (4) TNERPK[4]:  %5.3f        (10) TPFMQA[10]:
                %5.3f\n",fptr[9],fptr[21]);
printf("                             (5) TNERPK[5]:  %5.3f       9.  DEVPRT:
                %5.3f\n",fptr[10],fptr[22]);
printf("                             (6) TNERPK[6]:  %5.3f      10.  DSIPTK:
                %5.2f\n\n",fptr[11],fptr[23]);
printf("                        11. Size of project (KDSI):  %d\n\n",KDSI[0]);
printf("                        12. EXIT and SAVE changes.\n\n");

/* allows user to select a variable using assigned number and */
/* change current value by displaying just the variable selected */
/* once the new value is entered fuction returns to the display */
/* screen for user to see changes and allow additional changes */

printf("     Enter number of parameter you wish to change:  ");
scanf("%d",&choice1);

        switch(choice1)
        {
        case 1:
                    clrscr();
            gotoxy(10,10);
            printf("Enter Initial Under Staffing Level Factor:  ");
            scanf("%f",&fptr[0]);
            break;
        case 2:
                clrscr();
            gotoxy(10,10);
            printf("Enter Average Daily Manpower per Staff Member:  ");
            scanf("%f",&fptr[1]);
            break;
        case 3:
            clrscr();
            gotoxy(10,10);
            printf("Enter Hiring Delay:  ");
            scanf("%f",&fptr[2]);
            break;
        case 4:
                    clrscr();
            gotoxy(10,10);
```

```
            printf("Enter Average Employment Time:  ");
            scanf("%f",&fptr[3]);
            break;
        case 5:
                    clrscr();
            gotoxy(10,10);
            printf("Enter Training Overhead:  ");
            scanf("%f",&fptr[4]);
            break;
        case 6:
                    clrscr();
            gotoxy(10,10);
            printf("Enter Average Assimilation Delay:  ");
            scanf("%f",&fptr[5]);
            break;

        /* The TNERPK has several entries for this one variable by using */
        /* a second set of values for each entry the user can change one */
        /* entry at a time vice entring all the values each time even if */
        /* one value needed to be changed.  */

        case 7:
            printf("     Enter subscript value of TNERPK parameter you wish
to change:  ");
            scanf("%d",&choice2);
            switch(choice2)
            {
                case 1:
                    clrscr();
                    gotoxy(10,10);
                    printf("Enter Error rate[1]: ");
                    scanf("%f",&fptr[6]);
                    break;
                case 2:
                    clrscr();
                    gotoxy(10,10);
                    printf("Enter Error rate[2]: ");
                    scanf("%f",&fptr[7]);
                    break;
                case 3:
                    clrscr();
                    gotoxy(10,10);
                    printf("Enter Error rate[3]: ");
                    scanf("%f",&fptr[8]);
                    break;
                case 4:
                    clrscr();
                    gotoxy(10,10);
                    printf("Enter Error rate[4]: ");
                    scanf("%f",&fptr[9]);
                    break;
                case 5:
                    clrscr();
                    gotoxy(10,10);
                    printf("Enter Error rate[5]: ");
                    scanf("%f",&fptr[10]);
                    break;
                case 6:
                    clrscr();
                    gotoxy(10,10);
                    printf("Enter Error rate[6]: ");
```

```
                    scanf("%f",&fptr[11]);
                    break;
                default:
                    break;
            }
            break;

        /* TPFMQA set-up same way as TNERPK for same reasons */
        case 8:
            printf("     Enter subscript value of TPFMQA parameter you
wish to change:  ");
            scanf("%d",&choice3);
            switch(choice3)
            {
            case 1:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[1]:  ");
                scanf("%f",&fptr[12]);
                break;
                    case 2:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[2]:  ");
                scanf("%f",&fptr[13]);
                break;
                    case 3:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[3]:  ");
                scanf("%f",&fptr[14]);
                break;
                    case 4:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[4]:  ");
                scanf("%f",&fptr[15]);
                break;
            case 5:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[5]:  ");
                scanf("%f",&fptr[16]);
                break;
            case 6:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[6]:  ");
                scanf("%f",&fptr[17]);
                break;
                    case 7:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[7]:  ");
                scanf("%f",&fptr[18]);
                break;
                    case 8:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[8]:  ");
                scanf("%f",&fptr[19]);
```

```
                break;
                    case 9:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[9]:  ");
                scanf("%f",&fptr[20]);
                break;
            case 10:
                clrscr();
                    gotoxy(10,10);
                printf("Enter Planned Fraction of Manpower for QA[10]: ");
                scanf("%f",&fptr[21]);
                break;
            default:
                break;

        }
            break;
        case 9:
                    clrscr();
            gotoxy(10,10);
            printf("Enter DEVPRT:  ");
            scanf("%f",&fptr[22]);
            break;
        case 10:
                    clrscr();
            gotoxy(10,10);
            printf("Enter Nominal Potential Productivity MD percent :
\n\n");
            printf("              Percent MD for tests:  ");
            scanf("%f",&PCNT[0]);
            printf("\n  Please be consistent with TPFMQA input values)\n");
            printf("              Percent MD for QA:     ");
            scanf("%f",&PCNT[1]);
            printf("              Percent MD for Rework: ");
            scanf("%f",&PCNT[2]);

            break;
        case 11:
            clrscr();
                    gotoxy(10,10);
            printf("Enter new Size of project (KDSI):  ");
            scanf("%d",&KDSI[0]);
            break;
        default:
            done1 = 1;
        }  /* switch choice 1 */
        switch(mode)
        {
        case 1:
            exp1 =  1.05;
            exp2 =  0.38;
                    calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
         case 2:
            exp1 =  1.12;
            exp2 =  0.35;
                    calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
         case 3:
            exp1 =  1.20;
```

```
                exp2 =  0.32;
                        calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
                break;
        }
        /* Nominal productivity is one of 3 variables passed into */
        /* the simulation model that need algorithmic calculations. */
        /* This function will be discussed in detail below */
        fptr[23] = prod(PCNT,results[0],results[2],KDSI);
    }  /* while done1 */
}

/* This function gives the user the opportunity to view current */
/* values assigned to the COCOMO 15 cost drivers and make changes */
/* if necessary.  All cost drivers are defaulted to 1.00. */

void icocomo_in(float *rate,float *EAF1,int done3,char *fname1)
{
  /* rate is an array which hold the values for determining EAF for COCOMO */
    int choice,i;
    int CD1,CD2,CD3,CD4,CD5,CD6,CD7,CD8,CD9,CD10,CD11,CD12,CD13,CD14,CD15;

  /* clears screen and displays the 15 cost drivers and values */
   while(!done3)
   {
   clrscr();
   printf("           ********************************************\n");
   printf("               INTERMEDIATE LEVEL COCOMO MODEL INPUTS   \n");
   printf("                           for %s                \n",fname1);
   printf("           ********************************************\n\n");
   printf("                        1.  RELY:  %1.2f\n",rate[0]);
   printf("                        2.  DATA:  %1.2f\n",rate[1]);
   printf("                        3.  CPLX:  %1.2f\n",rate[2]);
   printf("                        4.  TIME:  %1.2f\n",rate[3]);
   printf("                        5.  STOR:  %1.2f\n",rate[4]);
   printf("                        6.  VIRT:  %1.2f\n",rate[5]);
   printf("                        7.  TURN:  %1.2f\n",rate[6]);
   printf("                        8.  ACAP:  %1.2f\n",rate[7]);
   printf("                        9.  AEXP:  %1.2f\n",rate[8]);
   printf("                       10.  PCAP:  %1.2f\n",rate[9]);
   printf("                       11.  VEXP:  %1.2f\n",rate[10]);
   printf("                       12.  LEXP:  %1.2f\n",rate[11]);
   printf("                       13.  MODP:  %1.2f\n",rate[12]);
   printf("                       14.  TOOL:  %1.2f\n",rate[13]);
   printf("                       15.  SCED:  %1.2f\n\n",rate[14]);
   printf("                            16.   Press [16 or 0] when entries are
                    complete.\n\n");

        /* allows user to select one of the above cost drivers by number */
        /* using the case statments the program calls specific pop-up */
        /* menus for the user to select specific values from and return */
        /* to display screen to see changes.  */
   printf("             Select Cost Driver and press [Enter]: ");
   scanf("%d",&choice);
   switch(choice)
   {
        case 1:
            CD1 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc1, TRUE);
            switch (CD1)
            {
                case 0:
                    break;
```

```
    case 1:
        rate[0] = 0.75;
        break;
    case 2:
        rate[0] = 0.88;
        break;
    case 3:
        rate[0] = 1.00;
        break;
    case 4:
        rate[0] = 1.15;
        break;
    case 5:
        rate[0] = 1.40;
        break;
    }
    break;
case 2:
    CD2 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc2, TRUE);
    switch (CD2)
    {
    case 0:
        break;

    case 1:
        rate[1] = 0.94;
        break;
    case 2:
        rate[1] = 1.00;
        break;
    case 3:
        rate[1] = 1.08;
        break;
    case 4:
        rate[1] = 1.16;
        break;
    }
       break;
case 3:
    CD3 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc3, TRUE);
    switch (CD3)
    {
    case 0:
        break;

    case 1:
        rate[2] = 0.70;
        break;
    case 2:
        rate[2] = 0.85;
        break;
    case 3:
        rate[2] = 1.00;
        break;
    case 4:
        rate[2] = 1.15;
        break;
    case 5:
        rate[2] = 1.30;
        break;
    case 6:
```

```
            rate[2] = 1.65;
            break;
        }
       break;
case 4:
    CD4 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc4, TRUE);
    switch (CD4)
    {
    case 0:
        break;

    case 1:
        rate[3] = 1.00;
        break;
    case 2:
        rate[3] = 1.11;
        break;
    case 3:
        rate[3] = 1.30;
        break;
    case 4:
        rate[3] = 1.66;
        break;
    }
              break;
case 5:
    CD5 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc5, TRUE);
    switch (CD5)
    {
    case 0:
        break;

    case 1:
        rate[4] = 1.00;
        break;
    case 2:
        rate[4] = 1.06;
        break;
    case 3:
        rate[4] = 1.21;
        break;
    case 4:
        rate[4] = 1.56;
        break;
    }
            break;
case 6:
    CD6 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc6, TRUE);
    switch (CD6)
    {
    case 0:
        break;

    case 1:
        rate[5] = 0.87;
        break;
    case 2:
        rate[5] = 1.00;
        break;
    case 3:
        rate[5] = 1.15;
```

```
        break;
    case 4:
        rate[5] = 1.30;
        break;
    }
    break;
case 7:
    CD7 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc7, TRUE);
    switch (CD7)
    {
    case 0:
        break;

    case 1:
        rate[6] = 0.87;
        break;
    case 2:
        rate[6] = 1.00;
        break;
    case 3:
        rate[6] = 1.07;
        break;
    case 4:
        rate[6] = 1.15;
        break;
    }
    break;
case 8:
    CD8 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc8, TRUE);
    switch (CD8)
    {
    case 0:
        break;

    case 1:
        rate[7] = 1.46;
        break;
    case 2:
        rate[7] = 1.19;
        break;
    case 3:
        rate[7] = 1.00;
        break;
    case 4:
        rate[7] = 0.86;
        break;
            case 5:
        rate[7] = 0.71;
        break;
    }
            break;
case 9:
    CD9 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc9, TRUE);
    switch (CD9)
    {
    case 0:
        break;

    case 1:
        rate[8] = 1.29;
        break;
```

```
    case 2:
        rate[8] = 1.13;
        break;
    case 3:
        rate[8] = 1.00;
        break;
    case 4:
        rate[8] = 0.91;
        break;
    case 5:
        rate[8] = 0.82;
        break;
    }
            break;
case 10:
    CD10 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc10, TRUE);
    switch (CD10)
    {
    case 0:
        break;

    case 1:
        rate[9] = 1.42;
        break;
    case 2:
        rate[9] = 1.17;
        break;
    case 3:
        rate[9] = 1.00;
        break;
    case 4:
        rate[9] = 0.86;
        break;
    case 5:
        rate[9] = 0.70;
        break;

    }
            break;
case 11:
    CD11 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc11, TRUE);
    switch (CD11)
    {
    case 0:
        break;

    case 1:
        rate[10] = 1.21;
        break;
    case 2:
        rate[10] = 1.10;
        break;
    case 3:
        rate[10] = 1.00;
        break;
    case 4:
        rate[10] = 0.90;
        break;
    }
            break;
case 12:
```

```
    CD12 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc12, TRUE);
    switch (CD12)
    {
    case 0:
        break;

    case 1:
        rate[11] = 1.14;
        break;
    case 2:
        rate[11] = 1.07;
        break;
    case 3:
        rate[11] = 1.00;
        break;
    case 4:
        rate[11] = 0.95;
        break;
    }
            break;
case 13:
    CD13 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc13, TRUE);
    switch (CD13)
    {

    case 0:
        break;

    case 1:
        rate[12] = 1.24;
        break;
    case 2:
        rate[12] = 1.10;
        break;
    case 3:
        rate[12] = 1.00;
        break;
    case 4:
        rate[12] = 0.91;
        break;
            case 5:
        rate[12] = 0.82;
        break;
    }
            break;
case 14:
    CD14 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc14, TRUE);
    switch (CD14)
    {
    case 0:
      - break;

    case 1:
        rate[13] = 1.24;
        break;
    case 2:
        rate[13] = 1.10;
        break;
    case 3:
        rate[13] = 1.00;
        break;
```

```
                case 4:
                    rate[13] = 0.91;
                    break;
                case 5:
                    rate[13] = 0.83;
                    break;
                }
                        break;

            /* for the schedule cost driver need both the EAF value */
            /* but also the actual percent of schedule compression */
            /* or expansion for later COCOMO calculations. */

            case 15:
                CD15 = wn_popup(0, 5, 15, 50, 10, wat, bat, &intelc15, TRUE);
                switch (CD15)
                {
                case 0:
                    break;

                case 1:
                    rate[14] = 1.23;
                    EAF1[0] = 0.75;
                    break;
                case 2:
                    rate[14] = 1.08;
                    EAF1[0] = 0.85;
                    break;
                case 3:
                    rate[14] = 1.00;
                    break;
                case 4:
                    rate[14] = 1.04;
                    EAF1[0] = 1.30;
                    break;
                case 5:
                    rate[14] = 1.10;
                    EAF1[0] = 1.60;
                    break;
                }
                break;
            default:
                done3 = 1;
                break;
            }

    }               /* while done3 */
}

/* This function saves all of the current data for each project */
/* under a specific name specified by the user */

void file_save(float *DSMI,float *cdrate,float *PCNT,char *fname1,float EAF,int
*KDSI,int mode, float mf1)
{


    if ((fout = fopen(fname1,"wb"))==NULL)
    {
        fprintf(stderr,"Unable to open file %s \n",fname1);
    }
```

```
   else
        {
   printf("\n\n\nSaving ..............%s\n\n\n\n",fname1);
   printf("\n\n\nSaving ..............%s\n",fname1);
   fwrite((void *) DSMI,26 * sizeof(float),1,fout);
   fwrite((void *) KDSI,sizeof(int),1,fout);
   fwrite((void *) &EAF,sizeof(float),1,fout);
   fwrite((void *) cdrate,15 * sizeof(float),1,fout);
   fwrite((void *) PCNT,3 * sizeof(float),1,fout);
   fwrite((void *) &mode,sizeof(int),1,fout);
   fwrite((void *) &mf1,sizeof(float),1,fout);
   fclose(fout);
   }
}

/* This function provides the avenue to interface the output */
/* variables from this program into the simulation model via a */
/* text file and pass certain other variables to the OUTPUT2 */
/* program via a binary file for reporting estimates and actual */
/* results and error rates. */

void file_pass(float TOTMD1,float TOTMD2,float TDEV1,float TDEV2,int *KDSI,int
*KDSI2,float *DSMI,float *DSMI2,float *PCNT,float *PCNT2)
{
   FILE *fpout;
   int loop_lim;
   float factor[7], factor2[4], TVALS[2], TVALS2[2];
   long l1,l2;

   /* need to change to long, once multiplied by 1000 */
   /* size could be out of integer range. */
   l1 = KDSI[0]*1000.0;
        l2 = KDSI2[0]*1000.0;

   /* factor and factor2 arrays contain inputs the user */
   /* enters to control adjustment factors in data and */
   /* the man-day percentage totals for testing, QA, and */
   /* rework to be passed to the iterative loop program */
   /* for use in updating the nominal productivity */

   factor[0] = (PCNT[0]+PCNT[1]+PCNT[2]);
   factor2[0] = (PCNT2[0]+PCNT2[1]+PCNT2[2]);
   KDSI[1] = 0;
   KDSI2[1]= 0;
   /* easier to pass as arrays */
   TVALS[0] = TOTMD1;
   TVALS[1] = TDEV1;
        TVALS2[0] = TOTMD2;
   TVALS2[1] = TDEV2;

   clrscr();
   gotoxy(10,3);
   printf("The following entries are START DATE and WORK FORCE CEILING
               entries\n");
   gotoxy(10,4);
   printf("for the purpose of several experiments that are outside the scope
of\n");
   gotoxy(10,5);
   printf("this thesis.\n");

   gotoxy(10,8);
```

```
printf("Enter the START DATE for project 1:  ");
scanf("%f",&factor[5]);
gotoxy(10,10);
printf("Enter the START DATE for project 2:  ");
scanf("%f",&factor2[3]);
gotoxy(10,12);
printf("Enter the WORK FORCE CEILING that applies to both projects:  ");
scanf("%f",&factor[6]);

/* Writes to textfile EXAMPLE.DNX */

fprintf(textfile,"C RJBDSI(1)=%ld\n", l1);
fprintf(textfile,"C RJBDSI(2)=%ld\n", l2);
fprintf(textfile,"C TOTMD1(1)=%5.0f\n", TVALS[0]);
fprintf(textfile,"C TOTMD1(2)=%5.0f\n", TVALS2[0]);
fprintf(textfile,"C TDEV1(1)=%5.0f\n", TVALS[1]);
fprintf(textfile,"C TDEV1(2)=%5.0f\n", TVALS2[1]);
fprintf(textfile,"C INUDST(1)=%5.2f\n", DSMI[0]);
fprintf(textfile,"C INUDST(2)=%5.2f\n", DSMI2[0]);
fprintf(textfile,"C ADMPPS(1)=%5.2f\n", DSMI[1]);
fprintf(textfile,"C ADMPPS(2)=%5.2f\n", DSMI2[1]);
fprintf(textfile,"C HIREDY(1)=%5.2f\n", DSMI[2]);
fprintf(textfile,"C HIREDY(2)=%5.2f\n", DSMI2[2]);
fprintf(textfile,"C AVEMPT(1)=%5.2f\n", DSMI[3]);
fprintf(textfile,"C AVEMPT(2)=%5.2f\n", DSMI2[3]);
fprintf(textfile,"C TRPNHR(1)=%5.2f\n", DSMI[4]);
fprintf(textfile,"C TRPNHR(2)=%5.2f\n", DSMI2[4]);
fprintf(textfile,"C ASIMDY(1)=%5.2f\n", DSMI[5]);
    fprintf(textfile,"C ASIMDY(2)=%5.2f\n", DSMI2[5]);
fprintf(textfile,"T   TNERP1=%5.2f   %5.2f   %5.2f   %5.2f   %5.2f   %5.2f\n",
        DSMI[6],DSMI[7],DSMI[8],DSMI[9],DSMI[10],DSMI[11]);
fprintf(textfile,"T   TNERP2=%5.2f   %5.2f   %5.2f   %5.2f   %5.2f   %5.2f\n",
        DSMI2[6],DSMI2[7],DSMI2[8],DSMI2[9],DSMI2[10],DSMI2[11]);
fprintf(textfile,"T TPFMQ1=%5.3f %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f
        %5.3f %5.3f 0\n", DSMI[12],DSMI[13],DSMI[14],DSMI[15],
        DSMI[16],DSMI[17],DSMI[18],DSMI[19],DSMI[20],DSMI[21]);
fprintf(textfile,"T TPFMQ2=%5.3f %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f
        %5.3f %5.3f 0\n", DSMI2[12],DSMI2[13],DSMI2[14],DSMI2[15],
        DSMI2[16],DSMI2[17],DSMI2[18],DSMI2[19],DSMI2[20],DSMI2[21]);
fprintf(textfile,"C DEVPRT(1)=%5.2f\n", DSMI[22]);
    fprintf(textfile,"C DEVPRT(2)=%5.2f\n", DSMI2[22]);
fprintf(textfile,"C DSIPTK(1)=%5.2f\n", DSMI[23]);
    fprintf(textfile,"C DSIPTK(2)=%5.2f\n", DSMI2[23]);
fprintf(textfile,"C STRTDT(1)=%5.2f\n", factor[5]);
fprintf(textfile,"C STRTDT(2)=%5.2f\n", factor2[3]);
fprintf(textfile,"C NCLTWF=%5.2f\n", factor[6]);
fclose(textfile);

/* The following user entries are requested here because */
/* it is just prior to running the simulation model.  */
/* They deal directly with the iterative loop process */
/* The requests dictate how accurate the user wishes to */
/* make the results, how many iterations of the loop should */
/* be run and an avenue for the user to build fat(safety factor) */
/* into the project or prevent the model from doing so. */

/* If the need for manual operation of the iterative loop */
/* process is necessary I would ask the user one additional */
/* question in this series. If the the user wished a manual or */
/* automatic loop process. Then label it as a flag and pass it */
/* in the binary file to the the testio2 program.  In that */
```

```
/* program is where I would create the manual system. */

clrscr();
gotoxy(10,3);
printf("The following entries are percentages used to prevent the model\n");
gotoxy(10,4);
printf("from building too much fat into the effort variable in the
project.\n");
gotoxy(10,5);
printf("In essence these factors simulate the managers responsibility not
to\n");
gotoxy(10,6);
printf("let the productivity lag.\n");
gotoxy(10,8);
printf("Enter the Effort adjustment factor in project 1 as a percent:  ");
scanf("%f",&factor[1]);
gotoxy(10,10);
printf("Enter the Effort adjustment factor in project 2 as a percent:  ");
scanf("%f",&factor2[1]);
gotoxy(10,12);
printf("Enter the Schedule adjustment factor in project 1 as a percent:  ");
scanf("%f",&factor[2]);
gotoxy(10,14);
     printf("Enter the Schedule adjustment factor in project 2 as a percent:
               ");
scanf("%f",&factor2[2]);
gotoxy(10,18);
printf("The following entries allow you to choose the accuracy level and\n");
gotoxy(10,19);
printf("limit the number of loops the model will run before completion.\n");
gotoxy(10,21);
printf("Enter the accuracy level for Effort as a percent:  ");
scanf("%f",&factor[3]);
gotoxy(10,23);
printf("Enter the accuracy level for Schedule as a percent:  ");
scanf("%f",&factor[4]);
gotoxy(10,25);
printf("Enter the limit of the maximum number of loops the model will do:
               ");
scanf("%d",&loop_lim);

/* Binary output file for use by output2 */

if ((fpout = fopen("outfile2.dnx","wb"))==NULL)
{
     fprintf(stderr,"Unable to open file %s \n","outfile2.dnx");
}
else
{
fwrite((void *) DSMI,26 * sizeof(float),1,fpout);
     fwrite((void *) DSMI2,26 * sizeof(float),1,fpout);
fwrite((void *) KDSI,2 * sizeof(int),1,fpout);
     fwrite((void *) KDSI2,2 * sizeof(int),1,fpout);
fwrite((void *) TVALS,2 * sizeof(float),1,fpout);
     fwrite((void *) TVALS2,2 * sizeof(float),1,fpout);
fwrite((void *) factor,7 * sizeof(float),1,fpout);
fwrite((void *) factor2,4 * sizeof(float),1,fpout);
fwrite((void *) &loop_lim,sizeof(int),1,fpout);
fclose(fpout);
}
```

```
}

/* Part of the calculation for Nominal Productivity requires */
/* interpolation.  This function accepts staff size variable */
/* and returns communication overhead factor for use in determining */
/* Nominal Productivity. */

float interp(float stf_size)
{
   float covhd;

   if ((stf_size >= 0) && (stf_size <= 5))
   {
        covhd = (((stf_size-0)* .015)/5);
   }
   if ((stf_size > 5) && (stf_size <= 10))
   {
        covhd = (((stf_size-5)* .045)/5) +.015;
   }
   if ((stf_size > 10) && (stf_size <= 15))
   {
        covhd = (((stf_size-10)* .075)/5) + .06;
   }
   if ((stf_size > 15) && (stf_size <= 20))
   {
        covhd = (((stf_size-15)* .105)/5) + .135;
   }
   if ((stf_size > 20) && (stf_size <= 25))
   {
        covhd = (((stf_size-20)* .135)/5)  + .24;
   }
   if ((stf_size > 25) && (stf_size <= 30))
   {
        covhd = (((stf_size-25)* .165)/5) + .375;
   }
   if (stf_size >= 30)
   {
        covhd = .54;
   }
   return covhd;
}

/* This function does the Nominal Productivity calculations */
/* TOTMD1 - Effort passed from main function in man-days */
/* TDEV2  - Schedule in months not days! */
/* PCNT   - array from main function which passes %Testing,%QA */
/*          and %Rework for man-days */
/* MM     - Effort in man-months */
/* stf_size - Average Staff Size = MM/TDEV2 */
/* DEVMD - Development man-days */
/* ADP    - Actual Development Productivity */
/* covhd - Communication Overhead */
/* product - Nominal Productivity */

float prod(float *PCNT,float TOTMD1,float TDEV2,int *KDSI)
{
        float MM,stf_size,DEVMD,ADP,covhd;
        float product;

        MM = TOTMD1/19;
        stf_size = MM/TDEV2;
```

```
        DEVMD = (1-(PCNT[0]+PCNT[1]+PCNT[2]))*TOTMD1;
        ADP = (KDSI[0] * 1000.0)/DEVMD;
        covhd = interp(stf_size);      /* call interpolation function */
        product = ADP/(0.6 * (1.0-covhd));
        return product;

}

/* This function completes COCOMO calculations for input into */
/* simulation model */
/* result array is used to hold man-day and schedule results */
/* EAF1 contains the percent to multiply TDEV by from cost driver 15 */
/* mf1, exp1 and exp2 are the coefficients and exponents passed */
/* in from main function */

void calc(float *result,int *KDSI,float *EAF1,float *cdrate,float mf1,float
exp1,float exp2)
{
   int i;
   float EAF;     /* Estimated Adjustment Factor */
   EAF = 1.00;
   for (i=0;i<15;i++)
   {
        EAF *= cdrate[i];    /* Calculate the EAF by multiplying each */
                             /* cost driver by one another */
        }
   /* Total man-days calculation */
   result[0] =  mf1 * (pow(KDSI[0],exp1)) * 19.0 * EAF;

   /* if cost driver 15 (schedule) is nominal then calculations */
   /* are straight forward. If not you must divide the man-days */
   /* by cdrate[14] or calculate total man-days as if schedule */
   /* was nominal. */

   if (EAF1[0] != 1.00)
   {
        result[1] =  2.5 * pow(((result[0]/19.0)/cdrate[14]),exp2) * EAF1[0] *
19.0;
        result[2] =  2.5 * pow(((result[0]/cdrate[14])/19.0),exp2) * EAF1[0];
   }
   else
   {
        result[1] =  2.5 * pow((result[0]/19.0),exp2) * 19.0;
        result[2] =  2.5 * pow((result[0]/19.0),exp2);
   }
   result[3] = EAF;
   return;
}

/* Small function that simply initializes all the */
/* cost drivers to 1 */

void initial(float *cdrate)
{
   int i;
   for(i=0;i < 15; i++)
   {
        cdrate[i] = 1.00;
   }
        return;
}
```

```
/* Fuction to accept initial project variable entries for simulation model */
void dsm_in(float *DSMI, float *PCNT)
{
   int i;

   /* Front end; allows user to make necessary inputs for the */
   /* Dynamic Simulation Model; Inputs appear one at a time and */
   /* there must be an entry for each variable; all inputs will */
   /* be stored in an array DSMI */
   clrscr();
   printf("                ****************************************\n");
   printf("                *   DYNAMIC SIMULATION MODEL INPUTS    *\n");
   printf("                ****************************************\n\n");
   printf("          Input the following: \n\n");
   printf("                1.  Initial Under Staffing Level Factor:  ");
   scanf("%f",&DSMI[0]);
   printf("                2.  Average Daily Manpower per Staff Member:  ");
   scanf("%f",&DSMI[1]);
   printf("                3.  Hiring Delay:  ");
   scanf("%f",&DSMI[2]);
   printf("                4.  Average Employment Time:  ");
   scanf("%f",&DSMI[3]);
   printf("                5.  Training Overhead:  ");
   scanf("%f",&DSMI[4]);
   printf("                6.  Average Assimilation Delay:  ");
   scanf("%f",&DSMI[5]);
   printf("                7.  Error Rate (Must enter 6 input values): \n");
   for(i=0; i < 6; i++)
   {
        printf("                          Error rate[%d]: ",(i+1));
        scanf("%f",&DSMI[i+6]);
   }
   printf("                8.  Planned Fraction of Manpower for QA \n");
   printf("                    (Must enter 10 input values):  \n");
   for(i=0; i < 10; i++)
   {
        printf("                           Manpower for QA[%d]: ",(i+1));
        scanf("%f",&DSMI[i+12]);
   }
   printf("                9.  DEVPRT:  ");
   scanf("%f",&DSMI[22]);
   printf("               10.  Nominal Potential Productivity Man Day percent
                          inputs:  \n\n");
      /* Following entries are necessary for above productivity calculation */
   printf("                          Percent MD for tests:  ");
   scanf("%f",&PCNT[0]);
      printf("\n              (Please be consistent with TPFMQA inputvalues)\n");
   printf("                          Percent MD for QA:     ");
   scanf("%f",&PCNT[1]);
   printf("                          Percent MD for Rework: ");
   scanf("%f",&PCNT[2]);
}

/* Fuction to accept initial project variable entries for COCOMO */
void como_in(int *KDSI,char *fname1)
{
   char string1[25];

   /* Allows user to make necessary inputs for the */
   /* COCOMO Model; Inputs appear one at a time and */
   /* there must be an entry for each variable */
```

```
   clrscr();
   printf("                ****************************************\n");
   printf("                *         COCOMO MODEL INPUTS          *\n");
      printf("                   ****************************************\n\n\n");
   printf("           Input the following: \n\n");
   printf("                1.  Estimated Project Size in KDSI:  ");
   scanf("%d",&KDSI[0]);   /* string gets KDSI value */

   printf("                2.  Enter the Project Name: ");
   scanf("%s",&string1);   /* string gets project name */

   strncpy(fname1, string1, 8); /* since dos only recognizes the first */
                          /* 8 characters fname1 takes first 8 */
                          /* characters in string1 */
   string1[8] = '\0';   /* resets string1 to null set so next */
                 /* project name if short will not contain */
                 /* characters previously resident in string1 */
   strcat(fname1,".prf");  /* automatically tags all project names */
                    /* with .prf to easily recognize projects */
}


void main()
{

   /* Declarations */

  int i, done=0,done1=0, done3=0, done5=0, done6=0, done7=0;
  int sel, sel1, sel2, sel3, sel4, sel5, sel6;
  int mode, mode2;
  int project1 = 0, project2 = 0;
  int KDSI[2], KDSI2[2];

  float EAF1[1], EAF2[1];
  float cdrate[15], DSMI[26], PCNT[3],results[4];
  float cdrate2[15], DSMI2[26], PCNT2[3],results2[4];
  float mf1, mf2, exp1, exp2;


  char fname1[13];
  char fname2[13];
  char newname[13];
  char string[25];
  char string1[25];

  int ch, basic;

  /* creates textfile which is interface with simulation model */
  textfile = fopen("simtwo.DNX","w");

  /* initializes scedule cost driver to 1 for both projects */
  EAF1[0] = 1.00;
  EAF2[0] = 1.00;

  /* bat is the boarder attribute for the pop-up window */
  /* sets background to blue and boarder to white */
  bat = v_setatr(BLUE,WHITE,0,0);

  /* wat is the window attribute for the pop-up window */
  /* sets background to blue and text to white */
```

```
 wat = v_setatr(BLUE,WHITE,0,0);

/* this while statement gets program started always initiated on */
while (!done6)
{
   clrscr();              /* pop-up initial menu */
                          /* allows user to go to main menu */
                          /* create a new project or EXIT */
  sel = wn_popup(0, 5, 10, 55, 10, wat, bat, &intelc, TRUE);

  switch (sel)
  {

  /* SELECT PROJECT FILES FROM DISK AND EXECUTE SIMULATION MODEL */
  case 1:     /* go to Main Menu */
   done5=0;
   while(!done5)
   {
      clrscr();              /* Main Menu will allow user to */
                  /* list, select, run simulation */
                  /* or exit this menu    */
     sel2 = wn_popup(0, 5, 10, 55, 12, wat, bat, &intelc0, TRUE);

     switch (sel2)
     {
        /*  PROJECT 1 RETRIVAL FROM DISK */
        case 1:
            filelist();   /* calls function to list all *.prf files */
                        /* user can look at list and enter file name */
            printf("\n\n:>  Enter project filename: ");
            scanf("%s",&fname1);
            /* if name of file is mis-entered program goes back to */
            /* main menu */
            if ((fin = fopen(fname1,"rb"))==NULL)
            {
               fprintf(stderr,"Unable to open file %s to read\n",fname1);
               continue;
            }
            /* read in the data of the selected filename */
            fread((void *) DSMI,26 * sizeof(float),1,fin);
            fread((void *) KDSI,sizeof(int),1,fin);
            fread((void *) &results[3],sizeof(float),1,fin);
            fread((void *) cdrate,15 * sizeof(float),1,fin);
            fread((void *) PCNT,3 * sizeof(float),1,fin);
            fread((void *) &mode,sizeof(int),1,fin);
            fread((void *) &mf1,sizeof(float),1,fin);
            fclose(fin);
            /* re-intialize */
            done1 = 0;
            project1 = 1;

            /* function described above which allows user to display */
            /* on screen the variables for the simulation model less */
            /* the COCOMO variables */
            model_in(DSMI,PCNT,KDSI,results,mode,
                        fname1,EAF1,cdrate,done1,mf1);

            if(results[3]==1.00)
            {
               clrscr();   /* gives user option to continue */
                           /* using basic model or use intermediate */
```

```
    sel3 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc21, TRUE);
    switch(sel3)
    {
        case 1:    /* user selected basic model */
            basic = 1;
            initial(cdrate);  /* Basic model EAF values must */
            break;            /* all be 1.00.  This sets all */
                  /* cost drivers to 1.00 */
        case 2:    /* user selected intermediate model */
            basic = 0;
            initial(cdrate);
           /* Displays cost driver screen; allows user to */
            /* set cost drivers to desired level */
            icocomo_in(cdrate,EAF1,done3,fname1);
            break;
    }  /* switch sel3 */
}  /* if */
else  /* if EAF is other than 1.00 */
{
    /* displays cost drivers values and */
    /* allows user to manipulate */
    basic = 0;
    icocomo_in(cdrate,EAF1,done3,fname1);
}
clrscr();
         /* pop-up menu for user to select COCOMO mode */
mode = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc20, TRUE);

switch (mode)
{
    case 1:              /* Organic mode */
        if (basic != 1)
        {
            mf1  =  3.2;    /* sets coefficient and exponents */
        }
        else
        {
            mf1  =  2.4;
        }
        exp1 =  1.05;   /* for man-days calculation */
        exp2 =  0.38;
        /* function that actually does the COCOMO calculations */
        calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
    case 2:              /* Semi-detached mode */
        mf1  =  3.0;
        exp1 =  1.12;
        exp2 =  0.35;
            calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
    case 3:              /* Embedded mode */
        if (basic != 1)
        {
            mf1  =  3.6;    /* sets coefficient and exponents */
        }
        else
        {
            mf1  =  2.8;
        }
        exp1 =  1.20;
        exp2 =  0.32;
```

```
                calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
            break;
    } /* switch mode */
   /* allows user to save current datafile under the same name */
    /* or save the same or manipulated data under a new name */
    sel4 = wn_popup(0, 5, 10, 55, 10, wat, bat, &intelc22, TRUE);

    switch (sel4)
    {
       case 1:  /* save in same file */
            file_save(DSMI,cdrate,PCNT,fname1,
                results[3],KDSI,mode,mf1);
            break;
       case 2:  /* if you are changing the name of the file */
            clrscr();
            gotoxy(12,10);   /* enter new name */
            printf("Enter the new project filename:  ");
            scanf("%s",&string);

       /* Program lets user enter more than 8 characters */
       /* for filename.  This copies first 8 characters */
       /* into nem filename variable */
           strncpy(fname2, string, 8);
       string[8] = '\0';         /* resets string to null */
       strcat(fname2,".prf");    /* automatically adds ".prf" */

                   /* writes new file to disk */
       fnew = fopen(fname2,"wb");
       fwrite((void *) DSMI,26 * sizeof(float),1,fnew);
       fwrite((void *) KDSI,sizeof(int),1,fnew);
       fwrite((void *) &results[3],sizeof(float),1,fnew);
       fwrite((void *) cdrate,15 * sizeof(float),1,fnew);
       fwrite((void *) PCNT,3 * sizeof(float),1,fnew);
       fwrite((void *) &mode,sizeof(int),1,fnew);
       fwrite((void *) &mf1,sizeof(float),1,fnew);
       fclose(fnew);
    }
    break;

/*  PROJECT 2 RETRIVAL FROM DISK */
case 2:
    filelist();     /* calls function to list all *.prf files */
                   /* user can look at list and enter file name */
    printf("\n\n:>  Enter project filename: ");
    scanf("%s",&fname2);

    /* if name of file is mis-entered program goes back to */
    /* main menu */
    if ((fin = fopen(fname2,"rb"))==NULL)
    {
       fprintf(stderr,"Unable to open file %s to read\n",fname2);
       continue;
    }
    /* read in the data of the selected filename */
    fread((void *) DSMI2,26 * sizeof(float),1,fin);
    fread((void *) KDSI2,sizeof(int),1,fin);
    fread((void *) &results2[3],sizeof(float),1,fin);
    fread((void *) cdrate2,15 * sizeof(float),1,fin);
    fread((void *) PCNT2,3 * sizeof(float),1,fin);
    fread((void *) &mode2,sizeof(int),1,fin);
    fread((void *) &mf2,sizeof(float),1,fin);
```

```
fclose(fin);
/* re-intialize */
done1 = 0;
    project2 = 1;

/* function described above which allows user to display */
/* on screen the variables for the simulation model less */
/* the COCOMO variables */
model_in(DSMI2,PCNT2,KDSI2,results2,mode2,
            fname2,EAF2,cdrate2,done1, mf2);

if(results2[3]==1.00)
{
   clrscr();  /* gives user option to continue */
            /* using basic model or use intermediate */
   sel3 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc21, TRUE);
   switch(sel3)
   {
       case 1:         /* user selected basic model */
           basic = 1;
           initial(cdrate2);  /* Basic model EAF values must */
           break;             /* all be 1.00.  This sets all */
           /* cost drivers to 1.00 */
       case 2:    /* user selected intermediate model */
           basic = 0;
           initial(cdrate2);
          /* Displays cost driver screen; allows user to */
       /* set cost drivers to desired level */
           icocomo_in(cdrate2,EAF2,done3,fname2);
           break;
   }  /* switch sel3 */
}  /* if */
else  /* if EAF is other than 1.00 */
{
   /* displays cost drivers values and */
   /* allows user to manipulate */
   basic = 0;
   icocomo_in(cdrate2,EAF2,done3,fname2);
}
clrscr();

         /* pop-up menu for user to select COCOMO mode */
mode2 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc20, TRUE);

switch (mode2)
{
   case 1:             /* Organic mode */
       if (basic != 1)
       {
           mf2  =  3.2;    /* sets coefficient and exponents */
       }
       else
       {
           mf2  =  2.4;
       }
       exp1 =  1.05;  /* for man-days calculation */
       exp2 =  0.38;
       /* function that actually does the COCOMO calculations */
       calc(results2,KDSI2,EAF2,cdrate2,mf2,exp1,exp2);
       break;
   case 2:             /* Semi-detached mode */
```

```
            mf2  =  3.0;
            exp1 =  1.12;
            exp2 =  0.35;
            calc(results2,KDSI2,EAF2,cdrate2,mf2,exp1,exp2);
            break;
        case 3:             /* Embedded mode */
            if (basic != 1)
            {
                mf2  =  3.6;    /* sets coefficient and exponents */
            }
            else
            {
                mf2  =  2.8;
            }
            exp1 =  1.20;
            exp2 =  0.32;
            calc(results2,KDSI2,EAF2,cdrate2,mf2,exp1,exp2);
            break;
     } /* switch mode */

    /* allows user to save current datafile under the same name */
    /* or save the same or manipulated data under a new name */
     sel4 = wn_popup(0, 5, 10, 55, 10, wat, bat, &intelc22, TRUE);

     switch (sel4)
     {
        case 1:  /* save in same file */
            file_save(DSMI2,cdrate2,PCNT2,fname2,
                results2[3],KDSI2,mode2,mf2);
            break;
        case 2:  /* if you are changing the name of the file */
            clrscr();
            gotoxy(12,10);   /* enter new name */
            printf("Enter the new project filename:  ");
            scanf("%s",&string);

        /* Program lets user enter more than 8 characters */
        /* for filename.  This copies first 8 characters */
        /* into nem filename variable */
                strncpy(newname, string, 8);
        string[8] = '\0';      /* resets string to null */
        strcat(newname,".prf"); /* automatically adds ".prf" */

                    /* writes new file to disk */
        fnew2 = fopen(newname,"wb");
        fwrite((void *) DSMI2,26 * sizeof(float),1,fnew2);
        fwrite((void *) KDSI2,sizeof(int),1,fnew2);
        fwrite((void *) &results2[3],sizeof(float),1,fnew2);
        fwrite((void *) cdrate2,15 * sizeof(float),1,fnew2);
        fwrite((void *) PCNT2,3 * sizeof(float),1,fnew2);
        fwrite((void *) &mode2,sizeof(int),1,fnew2);
        fwrite((void *) &mf2, sizeof(float),1,fnew2);
        fclose(fnew2);
     }
     break;

/* RUNS TWO PROJECT DYNAMIC SIMULATION */
case 3:

     /* This is an error checking routine to ensure */
     /* both projects are loaded before executing the */
```

```
      /* simulation model */
      if (project1)  /* project1 is a flag */
                /* when 1 project1 is loaded */
                /* when 0 project1 is not loaded */
      {
         if (project2)  /* project2 is a flag */
                   /* when 1 project2 is loaded */
                        /* when 0 project1 is not loaded */
         {
/* Both projects are loaded, save files and execute simulation */
              done5 = 1;
              done6 = 1;
              file_pass(results[0],results2[0],results[1],
              results2[1],KDSI,KDSI2,DSMI,DSMI2,PCNT,PCNT2);
         }
         else
         {
              /* Only project 1 is loaded, will not proceed */
              clrscr();
              gotoxy(10,10);
              printf("You have not selected Project 2.\n");
                  gotoxy(10,11);
              printf("Both   projects   must   be   selected   to   run
                  simulation.\n");
              gotoxy(10,15);
              printf("Press any key to continue.........\n");
              getch();
         }
      }
      else
      {
         if (project2)
         {
             /* Only project 2 is loaded, will not proceed */
                  clrscr();
              gotoxy(10,10);
              printf("You have not selected Project 1.\n");
                  gotoxy(10,11);
              printf("Both   projects   must   be   selected   to   run
                  simulation.\n");
              gotoxy(10,15);
              printf("Press any key to continue.........\n");
              getch();
         }
         else
         {
              /* Neither project is loaded, will not proceed */
              clrscr();
              gotoxy(10,10);
              printf("You have not selected Project 1 or Project 2.\n");
                  gotoxy(10,11);
              printf("Both   projects   must   be   selected   to   run
                  simulation.\n");
              gotoxy(10,15);
                  printf("Press any key to continue.........\n");
              getch();
         }
      }
      break;

   /* EXITS CURRENT MENU TO PREVIOUS MENU (INITIAL MENU) */
```

```
      case 4:
           done5 = 1;
           done6 = 0;
           break;
       }  /* sel2 */
 }  /* while done5 */
 break;

/* CREATE NEW PROJECT FILES AND EXECUTE SIMULATION MODEL WHEN COMPLETE */
case 2:
     wn_init();  /* initialize a window for text entry */
     w4 = wn_open(0,5,10,58,10,wat,bat); /* open window w4; similar */
                    /* to opening a file */
/* 5 is starting row; 10 is starting column; 58 is characters wide */
/* second 12 is number of rows; wat is the window attribute and */
/* bat the boarder attribute */
     if(!w4) exit(1);

wn_printf(w4,"  \n ******************* IMPORTANT ********************\n");
wn_printf(w4,"  \n\n  In order to load NEW projects you must enter\n");
wn_printf(w4,"  input data for both COCOMO and the Dynamic Simulation.\n");
wn_printf(w4,"  There are two forms on which all data must be entered.\n");
wn_printf(w4,"  Please enter the data as accurately as possible.\n\n\n\n");
wn_printf(w4,"            Press [ANY KEY] to continue...");

v_getch();
wn_close(w4);
done7=0;
while (!done7)
{
clrscr();
/* Pop-up window for New Project selection */
sel6 = wn_popup(0, 5, 15, 55, 10, wat, bat, &intelc25, TRUE);
switch(sel6)
{
     /* NEW PROJECT 1 ENTRIES */
     case 1:

          /* This function allows user to enter simulation variables */
              /* into two arrays for project 1 */
          dsm_in(DSMI,PCNT);

          /* This function allows user to enter COCOMO variables */
          /* into an array and name the new project1 file */
          como_in(KDSI,fname1);

          clrscr();
          sel1 = wn_popup(0, 5, 15, 55, 10, wat, bat, &intelc19, TRUE);
                        /* selection of basic or intermediate COCOMO */
          switch(sel1)
          {
          case 1:   /* user selection basic */
              basic = 1;
              EAF1[0] = 1.00;/* initializes schedule percent to 100 */
              initial(cdrate); /* sets all cost drivers to nominal */
              break;
          case 2:   /* user selection intermediate */
              basic = 0;
              done3 = 0;
              EAF1[0] = 1.00;/* initializes schedule percent to 100 */
              initial(cdrate);  /* sets all cost drivers to nominal */
```

```
    icocomo_in(cdrate,EAF1,done3,fname1); /* allows user to set */
                        /* cost driver values */
    break;
}
clrscr();

mode = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc20, TRUE);
                            /* select a mode */
switch (mode)
{
    case 1:     /* Organic */
        if (basic != 1)
        {
            mf1  =  3.2;    /* sets coefficient and exponents */
        }
        else
        {
            mf1  =  2.4;
        }
        exp1 =  1.05;       /* for man-days calculation */
        exp2 =  0.38;
        /* calls function to do COCOMO calculations */
            calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
    case 2:     /* Semi-detached */
        mf1  =  3.0;
        exp1 =  1.12;
        exp2 =  0.35;
        /* calls function to do COCOMO calculations */
            calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
    case 3:     /* Embedded */
        if (basic != 1)
        {
            mf1  =  3.6;    /* sets coefficient and exponents */
        }
        else
        {
            mf1  =  2.8;
        }
        exp1 =  1.20;
        exp2 =  0.32;
        /* calls function to do COCOMO calculations */
            calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
}
/* calls function to do nominal productivity calculations */
/* results[0]=Total man-days; results[2]=schedule in months */
DSMI[23] = prod(PCNT,results[0],results[2],KDSI);

/* display/edit simulation model inputs */
model_in(DSMI,PCNT,KDSI,results,mode,
            fname1,EAF1,cdrate,done1,mf1);

/* save updated file automatically */
file_save(DSMI,cdrate,PCNT,fname1,results[3],KDSI,mode,mf1);

project1 = 1; /* Flag when equal to 1 indicates project1
            loaded */

/* this ends the input phase and initial COCOMO
```

```
                calculations */
    /* for project1 the user will return to menu to select new */
    /* project2, to Display/Edit or to run simulation model */
    break;

/* NEW PROJECT 2 ENTRIES */
case 2:

    /* This function allows user to enter simulation variables */
    /* into two arrays for project 2 */
    dsm_in(DSMI2,PCNT2);

    /* This function allows user to enter COCOMO variables */
    /* into an array and name the new project2 file */
    como_in(KDSI2,fname2);

    clrscr();
    sel1 = wn_popup(0, 5, 15, 55, 10, wat, bat, &intelc19, TRUE);
           /* selection of basic or intermediate COCOMO */
    switch(sel1)
    {
    case 1:       /* user selection basic */
        basic = 1;
        EAF2[0] = 1.00;/* initializes schedule percent to 100 */
        initial(cdrate2); /* sets all cost drivers to nominal */
        break;
    case 2:       /* user selection intermediate */
        basic = 0;
        done3 = 0 ;
        EAF2[0] = 1.00;     /* initializes schedule percent to 100 */
        initial(cdrate2); /* sets all cost drivers to nominal */
        icocomo_in(cdrate2,EAF2,done3,fname2);/* allows user to set */
                    /* cost driver values */
        break;
    }
    clrscr();

    mode2 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc20, TRUE);
                                    /* select a mode */
    switch (mode2)
    {
        case 1:    /* Organic */
            if (basic != 1)
            {
                mf2  =  3.2;    /* sets coefficient and exponents */
            }
            else
            {
                mf2  =  2.4;
            }
            exp1 =  1.05;      /* for man-days calculation */
            exp2 =  0.38;
            /* calls function to do COCOMO calculations */
            calc(results2,KDSI2,EAF2,cdrate2,mf2,exp1,exp2);
            break;
        case 2:    /* Semi-detached */
            mf2  =  3.0;
            exp1 =  1.12;
            exp2 =  0.35;
            /* calls function to do COCOMO calculations */
            calc(results2,KDSI2,EAF2,cdrate2,mf2,exp1,exp2);
```

```
                break;
            case 3:     /* Embedded */
                if (basic != 1)
                {
                    mf2  =  3.6;     /* sets coefficient and exponents */
                }
                else
                {
                    mf2  =  2.8;
                }
                exp1 =  1.20;
                exp2 =  0.32;
                /* calls function to do COCOMO calculations */
                calc(results2,KDSI2,EAF2,cdrate2,mf2,exp1,exp2);
                break;
        }
    /* calls function to do nominal productivity calculations */
    /* results2[0]=Total man-days; results2[2]=schedule in months */
        DSMI2[23] = prod(PCNT2,results2[0],results2[2],KDSI2);

        /* display/edit simulation model inputs */
        model_in(DSMI2,PCNT2,KDSI2,results2,mode2,
                    fname2,EAF2,cdrate2,done1,mf2);

        /* save updated file automatically */
        file_save(DSMI2,cdrate2,PCNT2,fname2,
                    results2[3],KDSI2,mode2,mf2);

        project2 = 1; /* Flag when equal to 1 indicates project2
                    loaded */

        /* this ends the input phase and initial COCOMO
                    calculations */
        /* for project2 the user will return to menu to select new */
        /* project1, to Display/Edit or to run simulation model */
        break;

    /* GO TO NEXT LEVEL MENU TO DISPLAY PROJECTS OR RUN SIMULATION */
    case 3:

        done = 0;
        while(!done)    /* New Project Menu loop */
        {
            clrscr();
            sel5 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc23, TRUE);
                                /* New Project Menu */

            switch (sel5)
            {
            case 1:      /* user selected Display/Edit project 1*/
                          /* display or edit simulation model inputs */
                model_in(DSMI,PCNT,KDSI,results,mode,
                    fname1,EAF1,cdrate,done1,mf1);

                if(results[3]==1.00)  /* checks if EAF = 1.00 */
                {
                    clrscr();/* gives user option to continue */
                       /* using basic model or use intermediate */

                    sel3 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc21,
TRUE);
```

```
        switch(sel3)
        {
        case 1:      /* user selected basic model */
        basic = 1;
        initial(cdrate);/* Basic model EAF values must */
        break;      /* all be 1.00.  This sets all */
        /* cost drivers to 1.00 */
        case 2:      /* user selected intermediate model */
        basic = 0;
        initial(cdrate);   /* initialize cost drivers to 2 */

           /* Displays cost driver screen; allows user to */
    /* set cost drivers to desired level */
        icocomo_in(cdrate,EAF1,done3,fname1);
        break;
        }  /* switch sel3 */
    }  /* if */
    else
    /* if EAF is other than 1.00 */
    {
      /* displays cost drivers values and */
     /* allows user to manipulate */
        basic = 0;
        icocomo_in(cdrate,EAF1,done3,fname1);
    }
    clrscr();
    switch (mode)/* mode at this point was previously */
    /* selected and will now be routed to */
    /* for proper calcualations */
    {
    case 1:  /* Organic mode */
        if (basic != 1)
        {
        mf1  =  3.2;    /* sets coefficient and exponents */
        }
        else
        {
        mf1  =  2.4;
        }
        exp1 =  1.05; /* for man-days calculation */
        exp2 =  0.38;

    /* function that actually does the COCOMO calculations */
        calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
    case 2:   /* Semi-detached mode */
        mf1  =  3.0;
        exp1 =  1.12;
        exp2 =  0.35;
        calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
        break;
    case 3:   /* Embedded mode */
        if (basic != 1)
        {
        mf1  =  3.6;    /* sets coefficient and exponents */
        }
        else
        {
        mf1  =  2.8;
        }
        exp1 =  1.20;
```

```
                        exp2 =  0.32;
                        calc(results,KDSI,EAF1,cdrate,mf1,exp1,exp2);
                        break;
                    } /* switch mode */
                    /* automatically save latest changes to file */
                    file_save(DSMI,cdrate,PCNT,fname1,results[3],KDSI,mode,mf1);
                    break;

                case 2:       /* user selected Display/Edit project 2*/
                    /* display or edit simulation model inputs */

model_in(DSMI2,PCNT2,KDSI2,results2,mode2,fname2,EAF2,cdrate2,done1,mf2);

                    if(results2[3]==1.00)    /* checks if EAF = 1.00 */
                    {
                        clrscr();
                        /* gives user option to continue */
                        /* using basic model or use intermediate */

                        sel3 = wn_popup(0, 5, 10, 50, 10, wat, bat, &intelc21,
TRUE);
                        switch(sel3)
                        {
                        case 1:      /* user selected basic model */
                        basic = 1;
                        initial(cdrate2); /* Basic model EAF values must*/
                        break;            /* all be 1.00.  This sets all */
                          /* cost drivers to 1.00 */
                        case 2:       /* user selected intermediate model */
                        basic = 0;
                        initial(cdrate2);
                        /* Displays cost driver screen;
                        allows user to */
                            /* set cost drivers to desired level */
                        icocomo_in(cdrate2,EAF2,done3,fname2);
                        break;
                        }  /* switch sel3 */
                    }  /* if */
                    else  /* if EAF is other than 1.00 */
                    {
                        /* displays cost drivers values and */
                        /* allows user to manipulate */
                        basic = 0;
                        icocomo_in(cdrate2,EAF2,done3,fname2);
                    }
                    clrscr();
                    switch (mode2)
                    {
                    case 1:     /* Organic mode */
                        if (basic != 1)
                        {
                        mf2  =  3.2;    /* sets coefficient and exponents */
                        }
                        else
                        {
                        mf2  =  2.4;
                        }
                        exp1 =  1.05; /* for man-days calculation */
                        exp2 =  0.38;
                       /* function that actually does the COCOMO
                        calculations */
```

```
                    calc(results2,KDSI2,EAF2,
                    cdrate2,mf2,exp1,exp2);
                    break;
                case 2:     /* Semi-detached mode */
                    mf2  =  3.0;
                    exp1 =  1.12;
                    exp2 =  0.35;
                    calc(results2,KDSI2,EAF2,
                    cdrate2,mf2,exp1,exp2);
                    break;
                case 3:     /* Embedded mode */
                    if (basic != 1)
                    {
                    mf2  =  3.2;    /* sets coefficient and exponents */
                    }
                    else
                    {
                    mf2  =  2.4;
                    }
                    exp1 =  1.20;
                    exp2 =  0.32;
                    calc(results2,KDSI2,EAF2,
                    cdrate2,mf2,exp1,exp2);
                    break;
                } /* switch mode */
                /* automatically save latest changes to file */
                file_save(DSMI2,cdrate2,PCNT2,
                    fname2,results2[3],KDSI2,mode2,mf2);
                break;
            case 3:   /* user selected to run simulation model */
                if (project1)    /* If TRUE project 1 data is entered */
                {
                    if (project2) /* If TRUE project 2 data is entered */
                    {
                      done = 1; /* allows user to exit to initial menu */
                      done6 = 1;/* allows user to exit out of program */
                      done7 = 1;
                      /* calls file_pass which outputs both SIMTWO.DNX */
                      /* and OUTFILE2.DNX.  After completion of t  h  i  s
function */
                      /* exit program to DOS which calls Simulation Model
*/
                      file_pass(results[0],results2[0],results[1],
                    results2[1],KDSI,KDSI2,DSMI,DSMI2,
                    PCNT,PCNT2);
                    }
                    else  /* Project 1 is entered but project 2 is not */
                    {
                    /* prints message to screen */
                    clrscr();
                    gotoxy(10,10);
                    printf("You have not selected
                    Project 2.\n");
                       gotoxy(10,11);
                    printf("Both projects must be selected t  o     r  u  n
simulation.\n");
                    gotoxy(10,15);
                    printf("Press any key to continue.....\n");
                    getch();
                    }
                }
```

```
                    else   /* project1 not loaded */
                    {
                        if (project2)   /* If TRUE */
                        /* Project 2 is entered but project 1 is not */
                        {
                            /* prints message to screen */
                        clrscr();
                        gotoxy(10,10);
                        printf("You have not selected
                        Project 1.\n");
                        gotoxy(10,11);
                        printf("Both projects must be selected t  o     r  u  n
simulation.\n");
                        gotoxy(10,15);
                        printf("Press any key to continue.........\n");
                        getch();
                        }
                        else  /* neither project is loaded */
                        {
                            /* prints message to screen */
                        clrscr();
                        gotoxy(10,10);
                        printf("You have not selected
                        Project 1 or Project 2.\n");
                        gotoxy(10,11);
                        printf("Both projects must be selected t  o     r  u  n
simulation.\n");
                        gotoxy(10,15);
                        printf("Press any key to continue.........\n");
                        getch();
                        }
                    }
                        break;
                case 4:
                    done = 1;         /* allows user to exit program */
                    done7 = 1;
                    break;
                }
            }
            break;
        }
        } /* while done7 */
        break;
   case 3:
        exit(1);  /* Exits program to DOS command line */
   } /* sel */


  }  /* while done6 */
  exit(0);   /* Exits program and returns back to DOS batch file.
}

/* THE  END of this program not project */

}
```

APPENDIX E

```
/* ********************************************************** */
/* * Author: Richard W. Smith    Advisor:  Prof. Abdel-Hamid  * */
/* * Program: OUTPUT2              Lang: C                     * */
/* * Used Shareware <windows.h> in project environment       * */
/* ********************************************************** */

/* This is one of 5 programs written and interfaced with the */
/* Dynamic Simulation Model.  This particular program gathers  */
/* outfile information from several files to generate reports */
/* and create an iterative loop environment for studying refinement */
/* of cost estimation.  */

/* The following headers were used and needed to utilize the */
/* library functions used throughout this program. */

#include <stdio.h>
#include <math.h>
#include <conio.h>

void file_pas(float *,float *,int *,int *,float *,float *,float *,float *,int);
float prod(float,float,float,int *);
float interp(float);
void cktwo(float,float,float,float,float,float,float *);
void passtwo(float,float,float,float,float,float);

/* This function provides the avenue to interface the output */
/* variables from this program into the simulation model via a */
/* text file and pass certain other variables back to the OUTPUT2 */
/* program via a binary file for reporting estimates and actual */
/* results and error rates. */

void file_pas(float *TVALS,float *TVALS2,int *KDSI,int *KDSI2,float *DSMI,float
*DSMI2,float *factor,float *factor2,int loop_lim)
{
   FILE *fpout, *textfile;

   /* need to change to long, once multiplied by 1000 */
   /* size could be out of integer range. */
   long l1,l2;
   l1 = KDSI[0]*1000.0;
   l2 = KDSI2[0]*1000.0;

    /* input file to simulation model */
   textfile = fopen("SIMTWO.DNX","w");

   fprintf(textfile,"C RJBDSI(1)=%ld\n", l1);
   fprintf(textfile,"C RJBDSI(2)=%ld\n", l2);
   fprintf(textfile,"C TOTMD1(1)=%5.0f\n", TVALS[0]);
   fprintf(textfile,"C TOTMD1(2)=%5.0f\n", TVALS2[0]);
   fprintf(textfile,"C TDEV1(1)=%5.0f\n", TVALS[1]);
   fprintf(textfile,"C TDEV1(2)=%5.0f\n", TVALS2[1]);
   fprintf(textfile,"C INUDST(1)=%5.2f\n", DSMI[0]);
   fprintf(textfile,"C INUDST(2)=%5.2f\n", DSMI2[0]);
   fprintf(textfile,"C ADMPPS(1)=%5.2f\n", DSMI[1]);
```

```
    fprintf(textfile,"C ADMPPS(2)=%5.2f\n", DSMI2[1]);
    fprintf(textfile,"C HIREDY(1)=%5.2f\n", DSMI[2]);
    fprintf(textfile,"C HIREDY(2)=%5.2f\n", DSMI2[2]);
    fprintf(textfile,"C AVEMPT(1)=%5.2f\n", DSMI[3]);
    fprintf(textfile,"C AVEMPT(2)=%5.2f\n", DSMI2[3]);
    fprintf(textfile,"C TRPNHR(1)=%5.2f\n", DSMI[4]);
    fprintf(textfile,"C TRPNHR(2)=%5.2f\n", DSMI2[4]);
    fprintf(textfile,"C ASIMDY(1)=%5.2f\n", DSMI[5]);
    fprintf(textfile,"C ASIMDY(2)=%5.2f\n", DSMI2[5]);
    fprintf(textfile,"T  TNERP1=%5.2f  %5.2f  %5.2f  %5.2f  %5.2f  %5.2f\n",
         DSMI[6],DSMI[7],DSMI[8],DSMI[9],DSMI[10],DSMI[11]);
    fprintf(textfile,"T   TNERP2=%5.2f   %5.2f   %5.2f   %5.2f   %5.2f   %5.2f\n",
         DSMI2[6],DSMI2[7],DSMI2[8],DSMI2[9],DSMI2[10],DSMI2[11]);
    fprintf(textfile,"T TPFMQ1=%5.3f %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f
              %5.3f %5.3f %5.3f 0\n", DSMI[12],DSMI[13],DSMI[14],DSMI[15],
              DSMI[16],DSMI[17],DSMI[18],DSMI[19],DSMI[20],DSMI[21]);
    fprintf(textfile,"T TPFMQ2=%5.3f %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f
              %5.3f %5.3f %5.3f 0\n", DSMI2[12],DSMI2[13],DSMI2[14],
              DSMI2[15],DSMI2[16],DSMI2[17],DSMI2[18],DSMI2[19],
              DSMI2[20],DSMI2[21]);
    fprintf(textfile,"C DEVPRT(1)=%5.2f\n", DSMI[22]);
      fprintf(textfile,"C DEVPRT(2)=%5.2f\n", DSMI2[22]);
    fprintf(textfile,"C DSIPTK(1)=%5.2f\n", DSMI[23]);
      fprintf(textfile,"C DSIPTK(2)=%5.2f\n", DSMI2[23]);
    fprintf(textfile,"C STRTDT(1)=%5.2f\n", factor[5]);
    fprintf(textfile,"C STRTDT(2)=%5.2f\n", factor2[3]);
    fprintf(textfile,"C NCLTWF=%5.2f\n", factor[6]);
    fclose(textfile);

    /* Binary output file for use by output2 */

    if ((fpout = fopen("outfile2.dnx","wb"))==NULL)
    {
         fprintf(stderr,"Unable to open file %s \n","outfile2.dnx");
    }
    else
    {
    fwrite((void *) DSMI,26 * sizeof(float),1,fpout);
      fwrite((void *) DSMI2,26 * sizeof(float),1,fpout);
    fwrite((void *) KDSI,2 * sizeof(int),1,fpout);
      fwrite((void *) KDSI2,2 * sizeof(int),1,fpout);
    fwrite((void *) TVALS,2 * sizeof(float),1,fpout);
      fwrite((void *) TVALS2,2 * sizeof(float),1,fpout);
    fwrite((void *) factor,7 * sizeof(float),1,fpout);
    fwrite((void *) factor2,4 * sizeof(float),1,fpout);
    fwrite((void *) &loop_lim,sizeof(int),1,fpout);
    fclose(fpout);
    }

}

/* This function does the Nominal Productivity calculations */
/* TOTMD1 - Effort passed from main function in man-days */
/* TDEV2  - Schedule in months not days! */
/* PCNT   - array from main function which passes %Testing,%QA */
/*          and %Rework for man-days */
/* MM     - Effort in man-months */
/* stf_size - Average Staff Size = MM/TDEV2 */
/* DEVMD - Development man-days */
/* ADP    - Actual Development Productivity */
/* covhd - Communication Overhead */
```

```
/* product - Nominal Productivity */

float prod(float pcnt,float TOTMD1,float TDEV2,int *KDSI)
{
        float MM,stf_size,DEVMD,ADP,covhd;
        float product;

        MM = TOTMD1/19;
        stf_size = MM/TDEV2;
        DEVMD = (1 - pcnt)*TOTMD1;
        ADP = (KDSI[0] * 1000.0)/DEVMD;
        covhd = interp(stf_size);       /* call interpolation function */
        product = ADP/(0.6 * (1.0-covhd));
        return product;

}

/* Part of the calculation for Nominal Productivity requires */
/* interpolation.  This function accepts staff size variable */
/* and returns communication overhead factor for use in determining */
/* Nominal Productivity. */

float interp(float stf_size)
{
   float covhd;

   if ((stf_size >= 0) && (stf_size <= 5))
   {
        covhd = (((stf_size-0)* .015)/5);
   }
   if ((stf_size > 5) && (stf_size <= 10))
   {
        covhd = (((stf_size-5)* .045)/5) +.015;
   }
   if ((stf_size > 10) && (stf_size <= 15))
   {
        covhd = (((stf_size-10)* .075)/5) + .06;
   }
   if ((stf_size > 15) && (stf_size <= 20))
   {
        covhd = (((stf_size-15)* .105)/5) + .135;
   }
   if ((stf_size > 20) && (stf_size <= 25))
   {
        covhd = (((stf_size-20)* .135)/5)  + .24;
   }
   if ((stf_size > 25) && (stf_size <= 30))
   {
        covhd = (((stf_size-25)* .165)/5) + .375;
   }
   if (stf_size >= 30)
   {
        covhd = .54;
   }
   return covhd;
}

/* This function checks Project 2 error rates to determine if */
/* error rates are within limits.  Only called if Project 1 */
/* meets its error rate requirements.  Otherwise goes to passtwo */
```

```
void cktwo(float TOTMD2,float TDEV2,float md2,float time2,float oldprod,float
           newprod,float *factor)
{
        float diff,diff1;
        int num;
        FILE *results;

        if (TOTMD2 >= md2)  /* checks if effort estimates are greater */
                            /* than actuals */
        {
            if (TDEV2 >= time2) /* checks if sked estimates are greater */
                            /* than actuals */
            {
                diff = (TOTMD2-md2)/md2;   /* error rate for effort */
                diff1 = (TDEV2-time2)/time2;/* error rate for sked */

                results = fopen("REPORT.OUT","a");  /* open file */
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD2   CUMMD2   ERROR   TDEV2       TIME2
ERROR    OLD     NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f   %6.2f
%6.2f %6.2f\n",TOTMD2,md2,diff,
                          TDEV2,time2,diff1,oldprod,newprod);
                fclose(results);

                /* check to see if error rates meet requiremnets */
                if  ((diff == 0 && diff1 <= factor[4]) ||
                     (diff <= factor[3] && diff1 == 0) ||
                     (diff <= factor[3] && diff1 <= factor[4]))
                {
                /* opens results file to print out at bottom of */
                /* report.out a reminder of availability of */
                /* output data */

                     results = fopen("REPORT.OUT","a");
                     fprintf(results,"\n\n**** This data is available i      n
REPORT.OUT ****\n");
                     fprintf(results,"**** Each time the model is run
REPORT.OUT will change ****\n");
                    fclose(results);

                     /* sets cursor at (col,row) */
                     /* Format for output of data */

                     gotoxy(10,10);
                     printf("      REPORT FORMAT CHOICE\n");
                     gotoxy(10,12);
                     printf("       1 - Display results\n");
                     gotoxy(10,13);
                     printf("       2 - Print results\n");
                     gotoxy(10,14);
                     printf("       3 - Exit\n");
                     gotoxy(10,16);
                     printf("Enter one of the above:  ");
                     scanf("%d",&num);

                     switch(num)
                     {
                         case 1:
                         clrscr();
```

```
                        exit(4);    /* sends to screen via DOS */
                        case 2:
                        clrscr();
                        exit(3);    /* sends to printer via DOS */
                        case 3:
                        exit (0);  /* exits program, but output */
                           /* still available in report.out file */
                    }
                }
            }
            else  /* case for effort estimate > actual, but sked */
                  /* actual > estimate */
            {
                diff = (TOTMD2-md2)/md2;    /* error rate for effort */
                diff1 = (time2-TDEV2)/time2; /* error rate for sked */
                /* open output file */
                results = fopen("REPORT.OUT","a");
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD2   CUMMD2   ERROR   TDEV2      TIME2
ERROR    OLD     NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TOTMD2,md2,diff,
                        TDEV2,time2,diff1,oldprod,newprod);
                fclose(results);

                /* check to see if error rates meet requiremnets */
                if ((diff == 0 && diff1 <= factor[4]) ||
                    (diff <= factor[3] && diff1 == 0) ||
                    (diff <= factor[3] && diff1 <= factor[4]))
                {
                /* opens results file to print out at bottom of */
                /* report.out a reminder of availability of */
                /* output data */
                    results = fopen("REPORT.OUT","a");
                    fprintf(results,"\n\n**** This data is available i      n
REPORT.OUT ****\n");
                    fprintf(results,"**** Each time the model is run REPORT.OUT
will change ****\n");
                    fclose(results);

                    /* sets cursor at (col,row) */
                    /* Format for output of data */
                    gotoxy(10,10);
                    printf("      REPORT FORMAT CHOICE\n");
                    gotoxy(10,12);
                    printf("       1 - Display results\n");
                    gotoxy(10,13);
                    printf("       2 - Print results\n");
                    gotoxy(10,14);
                    printf("       3 - Exit\n");
                    gotoxy(10,16);
                    printf("Enter one of the above:  ");
                    scanf("%d",&num);

                    switch(num)
                    {
                        case 1:
                           clrscr();
                        exit(4);    /* sends to screen via DOS */
                        case 2:
```

```
                        clrscr();
                    exit(3);   /* sends to printer via DOS */
                    case 3:
                    exit (0);  /* exits program, but output */
                       /* still available in report.out file */
                }
            }
        }
    }
    else   /* actual effort is > estimate effort */
    {
        if (TDEV2 >= time2) /* checks if sked estimates
                    are greater */
                       /* than actuals */
        {
            diff = (md2-TOTMD2)/md2;  /* error rate for effort */
            diff1 = (TDEV2-time2)/time2; /* error rate for sked */
            /* open output file */
            results = fopen("REPORT.OUT","a");
            fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
            fprintf(results,"TOTMD2   CUMMD2   ERROR   TDEV2   TIME2  ERROR
  OLD    NEW\n");
            fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TOTMD2,md2,diff,
                    TDEV2,time2,diff1,oldprod,newprod);
            fclose(results);

            /* check to see if error rates meet requiremnets */
            if ((diff == 0 && diff1 <= factor[4]) ||
                (diff <= factor[3] && diff1 == 0) ||
                (diff <= factor[3] && diff1 <= factor[4]))
            {
            /* opens results file to print out at bottom of */
            /* report.out a reminder of availability of */
            /* output data */
                results = fopen("REPORT.OUT","a");
                fprintf(results,"\n\n**** This data is available i      n
REPORT.OUT ****\n");
                fprintf(results,"**** Each time the model is run REPORT.OUT
will change ****\n");
                fclose(results);

                /* sets cursor at (col,row) */
                /* Format for output of data */
                gotoxy(10,10);
                printf("      REPORT FORMAT CHOICE\n");
                gotoxy(10,12);
                printf("       1 - Display results\n");
                gotoxy(10,13);
                printf("       2 - Print results\n");
                gotoxy(10,14);
                printf("       3 - Exit\n");
                gotoxy(10,16);
                printf("Enter one of the above:  ");
                scanf("%d",&num);

                switch(num)
                {
                    case 1:
                       clrscr();
```

```
                        exit(4);  /* sends to screen via DOS */
                        case 2:
                           clrscr();
                        exit(3);  /* sends to printer via DOS */
                        case 3:
                        exit (0); /* exits program, but output */
                           /* still available in report.out file */
                     }
                  }
               }
               else   /* actual effort and actual sked are > estimates */
               {
                    diff = (md2-TOTMD2)/md2;   /* error rate for effort */
                    diff1 = (time2-TDEV2)/time2; /* error rate for sked */
                    /* open file */
                    results = fopen("REPORT.OUT","a");
                    fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                    fprintf(results,"TOTMD2   CUMMD2   ERROR   TDEV2   TIME2  ERROR
  OLD    NEW\n");
                    fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TOTMD2,md2,diff,
                           TDEV2,time2,diff1,oldprod,newprod);
                    fclose(results);
                    /* check to see if error rates meet requiremnets */
                    if ((diff == 0 && diff1 <= factor[4]) ||
                        (diff <= factor[3] && diff1 == 0) ||
                        (diff <= factor[3] && diff1 <= factor[4]))
                    {
                    /* opens results file to print out at bottom of */
                    /* report.out a reminder of availability of */
                    /* output data */
                        results = fopen("REPORT.OUT","a");
                        fprintf(results,"\n\n**** This data is available i      n
REPORT.OUT ****\n");
                        fprintf(results,"**** Each time the model is run REPORT.OUT
will change ****\n");
                       fclose(results);

                        /* sets cursor at (col,row) */
                        /* Format for output of data */
                        gotoxy(10,10);
                        printf("      REPORT FORMAT CHOICE\n");
                        gotoxy(10,12);
                        printf("       1 - Display results\n");
                        gotoxy(10,13);
                        printf("       2 - Print results\n");
                        gotoxy(10,14);
                        printf("       3 - Exit\n");
                        gotoxy(10,16);
                        printf("Enter one of the above:  ");
                        scanf("%d",&num);

                        switch(num)
                        {
                            case 1:
                                      clrscr();
                            exit(4);  /* sends to screen via DOS */
                            case 2:
                               clrscr();
                            exit(3);  /* sends to printer via DOS */
```

```
                        case 3:
                        exit (0); /* exits program, but output */
                           /* still available in report.out file */
                    }
                }
            }

        }
}

/* This function checks Project 2 error rates to determine if */
/* error rates are within limits.  Only called if Project 1 */
/* does not meet its error rate requirements. Does not exit program */
/* from this function */

void passtwo(float TOTMD2,float TDEV2,float md2,float time2,float oldprod,float
newprod)
{
        float diff,diff1;
        FILE *results;

        if (TOTMD2 >= md2)        /* checks if effort estimates are greater */
        {                   /* than actuals */

            if (TDEV2 >= time2)    /* checks if sked estimates are greater */
            {                 /* than actuals */

                diff = (TOTMD2-md2)/md2;    /* error rate for effort */
                diff1 = (TDEV2-time2)/time2; /* error rate for sked */
                results = fopen("REPORT.OUT","a");
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD2   CUMMD2   ERROR   TDEV2   TIME2  ERROR
  OLD    NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TOTMD2,md2,diff,
                        TDEV2,time2,diff1,oldprod,newprod);
                fclose(results);
            }
            else  /* effort estimate > actual, but sked actual > estimate */
            {
                diff = (TOTMD2-md2)/md2;  /* error rate for effort */
                diff1 = (time2-TDEV2)/time2; /* error rate for sked */
                results = fopen("REPORT.OUT","a");
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD2   CUMMD2   ERROR   TDEV2   TIME2  ERROR
  OLD    NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TOTMD2,md2,diff,
                        TDEV2,time2,diff1,oldprod,newprod);
                fclose(results);
            }
        }
        else   /* effort actual > estimate */
        {
            if (TDEV2 >= time2)    /* checks if sked estimates are greater */
            {                 /* than actuals */

                diff = (md2-TOTMD2)/md2;    /* error rate for effort */
                diff1 = (TDEV2-time2)/time2;/* error rate for sked */
```

```
                results = fopen("REPORT.OUT","a");
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD2   CUMMD2   ERROR   TDEV2   TIME2  ERROR
  OLD    NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TOTMD2,md2,diff,
                        TDEV2,time2,diff1,oldprod,newprod);
                fclose(results);
            }
            else   /* actuals for effort and sked are > estimates */
            {
                diff = (md2-TOTMD2)/md2;  /* error rate for effort */
                diff1 = (time2-TDEV2)/time2;/* error rate for sked */
                results = fopen("REPORT.OUT","a");
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD2   CUMMD2   ERROR   TDEV2   TIME2  ERROR
  OLD    NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TOTMD2,md2,diff,
                        TDEV2,time2,diff1,oldprod,newprod);
                fclose(results);
            }

        }
}


void main(void)
{
   /* Declarations for variables used within this program */
   int i, k=0, m=0, n=0, p=0, KDSI[2], KDSI2[2];
   int count=0, loop_lim, num;
   float j, md, time, TOTMD1, TDEV1, DSMI[26], diff, diff1, pcnt;
   float md2, time2, TOTMD2, TDEV2, DSMI2[26], diff2, diff3, pcnt2;
   float oldprod, newprod,oldprod2, newprod2;
   float TVALS[2],TVALS2[2],factor[7],factor2[4];
   char string[12], string2[12], strng[12], strng2[12];
   char dest[12], dest1[12], dest2[12], dest3[12];
   FILE *fpin, *fdata, *results;

   /* initializes the all of the destination strings as null */
   for (i=0;i<12;i++)
   {
        dest[i]    = '\0';
        dest1[i]   = '\0';
        dest2[i]   = '\0';
        dest3[i]   = '\0';
        string[i]  = '\0';
        string2[i] = '\0';
        strng[i]   = '\0';
        strng2[i]  = '\0';
    }

   /* Read the outfile.dnx:  Binary file used in reporting */
   /* Easier to work with binary in this case */

   if ((fdata = fopen("outfile2.dnx","rb"))==NULL)
   {
```

```
     fprintf(stderr,"Unable to open file %s \n","outfile2.dnx");
}
else
{
fread((void *) DSMI,26 * sizeof(float),1,fdata);
fread((void *) DSMI2,26 * sizeof(float),1,fdata);
fread((void *) KDSI,2 * sizeof(int),1,fdata);
fread((void *) KDSI2,2 * sizeof(int),1,fdata);
fread((void *) TVALS,2 * sizeof(float),1,fdata);
fread((void *) TVALS2,2 * sizeof(float),1,fdata);
fread((void *) factor,7 * sizeof(float),1,fdata);
fread((void *) factor2,4 * sizeof(float),1,fdata);
fread((void *) &loop_lim,sizeof(int),1,fdata);
fclose(fdata);
}

fpin = fopen("SIMTWO.OUT","r");  /* get output from simulation */
                  /* Project 1 */
     /* GET EFFORT VALUE FROM SIMULATION OUTPUT FILE */
i = fgetc(fpin); /* get first character from output file */

while (i != 40)  /* continue getting characters until ascii */
                    /* #40 '(' */
{
     i = fgetc(fpin);
}
i = fgetc(fpin);
while (i != 41)     /* now get each char and save as string */
{
     string[k] = i;
     k++;               /* fill up string project 1 effort */
     i = fgetc(fpin);

}
string[k]='\0';
i = fgetc(fpin);

            /* GET EFFORT VALUE FROM SIMULATION OUTPUT FILE */
while (i != 40)
{
     i = fgetc(fpin);
}
i = fgetc(fpin);
while (i != 41)       /* continue getting characters until ascii */
                       /* #40 '(' */
{
     string2[m] = i;
     m++;                 /* fill up string project 2 effort */
     i = fgetc(fpin);
}
string2[m] = '\0';
i = fgetc(fpin);

/* GET SCHEDULE VALUE FROM SIMULATION OUTPUT FILE */
while (i != 40)      /* continue getting characters until ascii */
                      /* #40 '(' */
{
     i = fgetc(fpin);
}
i = fgetc(fpin);
while (i != 41)      /* continue getting characters until ascii */
```

```
                        /* #41 ')' */
{
     strng[n] = i;
     n++;                /* fill up string project 1 sked */
     i = fgetc(fpin);
}
strng[n] = '\0';  /* null */

i = fgetc(fpin);

 /* GET SCHEDULE VALUE FROM SIMULATION OUTPUT FILE */
while (i != 40)      /* continue getting characters until ascii */
                      /* #40 '(' */
{
     i = fgetc(fpin);
}
i = fgetc(fpin);
while (i != 41)       /* continue getting characters until ascii */
                       /* #41 ')' */
{
     strng2[p] = i;
     p++;                  /* fill up string project 2 sked */
     i = fgetc(fpin);
}
strng2[p] = '\0';
fclose(fpin);

strncpy(dest,string,k);  /* copy actual proj 1 effort into dest string */
md = atof(dest);         /* string to float conversion */
strncpy(dest1,string2,m);/* copy actual proj 2 effort into dest1 string */
md2 = atof(dest1);       /* string to float conversion */
strncpy(dest2,strng,n);  /* copy actual proj 1 sked into dest2 string */
time = atof(dest2);      /* string to float conversion */
strncpy(dest3,strng2,p); /* copy actual proj 2 sked into dest3 string */
time2 = atof(dest3);     /* string to float conversion */

oldprod = DSMI[23];     /* changes productivity variable name for */
   oldprod2 = DSMI2[23];   /* reporting purposes */

/* Calculate the new productivity values which change dynamically */
/* with changes in effort and schedule */
DSMI[23] = prod(factor[0],md,(TVALS[1]/19.0),KDSI);
DSMI2[23] = prod(factor2[0],md2,(TVALS2[1]/19.0),KDSI2);

count = KDSI[1];       /* counter initialization */
while (count != loop_lim)  /* checks to see if count equals the */
{                          /* limit on number of loops */
                     /* loop_lim is inouted directly by user */

     if (TVALS[0] >= md)     /* checks if proj 1 effort estimates are */
     {                    /* greater than actuals */

          if (TVALS[1] >= time)  /* checks if proj 1 sked estimates are */
          {                      /* greater than actuals */

               diff = (TVALS[0]-md)/md;      /* calc error rates for effort */
               diff1 = (TVALS[1]-time)/time; /* calc error rates for sked */
               results = fopen("REPORT.OUT","a");

               /* output format */
```

```
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD1   CUMMD1   ERROR   TDEV1   TIME1  ERROR
  OLD    NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f %6.2f
%6.2f\n",TVALS[0],md,diff,
                        TVALS[1],time,diff1,oldprod,DSMI[23]);
                fclose(results);

                /* check to see if error rates meet requiremnets */
                if ((diff == 0 && diff1 <= factor[4]) ||
                    (diff <= factor[3] && diff1 == 0) ||
                    (diff <= factor[3] && diff1 <= factor[4]))
                {
                    /* if it meets requirements go to cktwo to check proj 2 */
                    cktwo(TVALS2[0],TVALS2[1],md2,
                        time2,oldprod2,DSMI2[23],factor);
                }
                else
                {
                  /* if it meets requirements go to passtwo to check proj 2 */
                    passtwo(TVALS2[0],TVALS2[1],md2,
                        time2,oldprod2,DSMI2[23]);
                }
            }
            else   /* effort estimates > actuals and */
            {      /* sked actuals > estimates */

                diff = (TVALS[0]-md)/md;      /* calc error rates for effort */
                diff1 = (time-TVALS[1])/time; /* calc error rates for sked */
                results = fopen("REPORT.OUT","a");

                /* output format */
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD1   CUMMD1   ERROR   TDEV1   TIME1  ERROR
  OLD    NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TVALS[0],md,diff,
                        TVALS[1],time,diff1,oldprod,DSMI[23]);
                fclose(results);

                /* check to see if error rates meet requiremnets */
                if ((diff == 0 && diff1 <= factor[4]) ||
                    (diff <= factor[3] && diff1 == 0) ||
                    (diff <= factor[3] && diff1 <= factor[4]))
                {
                    /* if it meets requirements go to cktwo to check proj 2 */
                    cktwo(TVALS2[0],TVALS2[1],md2,
                        time2,oldprod2,DSMI2[23],factor);
                }
                else
                {
            /* if it meets requirements go to passtwo to check proj 2 */
                    passtwo(TVALS2[0],TVALS2[1],md2,
                        time2,oldprod2,DSMI2[23]);
                }
            }
        }
        else  /* effort actuals > estimates */
        {
```

```
            if (TVALS[1] >= time)  /* checks if proj 1 sked estimates are */
            {                      /* greater than actuals */

                diff = (md-TVALS[0])/md;      /* calc error rates for effort */
                diff1 = (TVALS[1]-time)/time; /* calc error rates for sked */
                results = fopen("REPORT.OUT","a");

                /* output format */
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD1   CUMMD1   ERROR   TDEV1   TIME1   ERROR
  OLD    NEW\n");
                fprintf(results,"%6.2f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TVALS[0],md,diff,TVALS[1],
                        time,diff1,oldprod,DSMI[23]);
                fclose(results);

                /* check to see if error rates meet requiremnets */
                        if ((diff == 0 && diff1 <= factor[4]) ||
                            (diff <= factor[3] && diff1 == 0) ||
                                    (diff <= factor[3] && diff1 <= factor[4]))
                {
                    /* if it meets requirements go to cktwo to check proj 2 */
                    cktwo(TVALS2[0],TVALS2[1],md2,
                        time2,oldprod2,DSMI2[23],factor);
                }
                else  /* does not meet error rate requirements */
                {
                  /* if it meets requirements go to passtwo to check proj 2 */
                        passtwo(TVALS2[0],TVALS2[1],md2,
                        time2,oldprod2,DSMI2[23]);
                }
            }
            else  /* effort and sked actuals are > estimates */
            {
                diff = (md-TVALS[0])/md;      /* calc error rates for effort */
                diff1 = (time-TVALS[1])/time; /* calc error rates for sked */
                results = fopen("REPORT.OUT","a");

                /* output format */
                fprintf(results,"\n                    PERCENT
PERCENT  PRODUCTIVITY\n");
                fprintf(results,"TOTMD1   CUMMD1   ERROR   TDEV1   TIME1   ERROR
  OLD    NEW\n");
                fprintf(results,"%6.0f  %6.0f %6.2f   %6.0f  %6.0f %6.2f  %6.2f
%6.2f\n",TVALS[0],md,diff,
                        TVALS[1],time,diff1,oldprod,DSMI[23]);
                fclose(results);


          /* check to see if error rates meet requiremnets */
                if ((diff == 0 && diff1 <= factor[4]) ||
                    (diff <= factor[3] && diff1 == 0) ||
                    (diff <= factor[3] && diff1 <= factor[4]))
                {
                    /* if it meets requirements go to cktwo to check proj 2 */
                    cktwo(TVALS2[0],TVALS2[1],md2,
                        time2,oldprod2,DSMI2[23],factor);
                }
                else
```

```
                {
                  /* if it meets requirements go to passtwo to check proj 2 */
                    passtwo(TVALS2[0],TVALS2[1],md2,
                        time2,oldprod2,DSMI2[23]);
                }
            }

        }
    count++;          /* counts number of loops or iterations */
    KDSI[1] = count;     /* re-sets counter number for next run */
    KDSI2[1] = count;

    TVALS[0] = md*factor[1];        /* adjusts effort updates for proj 1 */
    TVALS[1] = time*factor[2];      /* adjusts sked updates for proj 1 */
    TVALS2[0] = md2*factor2[1];     /* adjusts effort updates for proj 2 */
    TVALS2[1] = time2*factor2[2];   /* adjusts sked updates for proj 2 */

    file_pas(TVALS,TVALS2,KDSI,KDSI2,DSMI,DSMI2,factor,factor2,loop_lim);

        /* checks loop_lim against count to see if done */
        if (count != loop_lim)
        {
            exit (1);   /* exits program for next iteration */
        }
        else  /* count equaled loop_lim and is ready to exit program */
        {

        /* opens results file to print out at bottom of */
        /* report.out a reminder of availability of */
        /* output data */
        results = fopen("REPORT.OUT","a");
        fprintf(results,"\n\n**** This data is available in REPORT.OUT ****\n");
        fprintf(results,"**** Each time the model is run REPORT.OUT will change
****\n");
        fclose(results);
        }  /* if */
    }  /* while */

    /* sets cursor at (col,row) */
    /* Format for output of data */
    gotoxy(10,10);
    printf("      REPORT FORMAT CHOICE\n");
    gotoxy(10,12);
    printf("       1 - Display results\n");
    gotoxy(10,13);
    printf("       2 - Print results\n");
    gotoxy(10,14);
    printf("       3 - Exit\n");
    gotoxy(10,16);
    printf("Enter one of the above:  ");
    scanf("%d",&num);

    switch(num)
    {
        case 1:
            clrscr();
            exit(4);  /* sends to screen via DOS */
        case 2:
            clrscr();
            exit(3);  /* sends to printer via DOS */
        case 3:
```

```
                exit (0); /* exits program, but output */
                          /* still available in report.out file */
    }

}

/* end program */
```

LIST OF REFERENCES

ABDEL-HAMID, T. K. AND S. E. MADNICK, "Lessons Learned from Modeling the Dynamics of Software Development," Communications of the ACM, 32, 12, December (1989), 1426-1438.

ABDEL-HAMID, T. K. AND S. E. MADNICK, Software Project Dynamics: An Integrated Approach, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991.

ABDEL-HAMID, T. K. AND S. E. MADNICK, Software Development Dynamics: An Integrated Approach, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1990.

AGAN, C. E., Coupling Artificial Inteligence and a System Dynamics Simulation to Optimize Quality Assurance and Testing in Software Development, Thesis, Naval Postgraduate School, 1990.

BOEHM, B. W., "Improving Software Productivity," Computer (September 1987), 43-57. , Survey & Tutorial Series.

BOEHM, B. W., "Software Engineering Economics," IEEE Transactions on Software Engineering, SE-10, 1, January (1984), 4-21.

BOEHM, B. W., Software Engineering Economics, ed. R. T. Yeh,, Pretntice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

GHEZZI, C., M. JAZAYERI, AND D. MANDRIOLI, Fundamentals of Software Engineering, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991.

KITFIELD, J., "Is Software DOD's Achillies' Heel?," Military Forum (July 1989), 28-35.

PRESSMAN, R. S., Software Engineering: A Practitioner's Approach, 2nd ed., Series in Software Engineering and Technology, McGraw-Hill, Inc., 1987.

SCHLENDER, B. R., "How to Break the Software Logjam," Fortune (25 September 1989), 100-112.

UNITED STATES. CONGRESS. HOUSE. COMMITTEE ON GOVERNMENT OPERATIONS, DOD Automated Systems Experience Runaway Costs and Years of Schedule Delays While Providing Little Capability, H.Rept 382, 101st Cong., 1st sess., GPO, Washington, DC, 1989.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Dr. Tarek K. Abdel-Hamid, Code AS/AH Department of Administrative Sciences Naval Postgraduate School Monterey, CA 93943-5000	5
4.	Dr. Magdi Kamel, Code AS/KA Department of Administrative Sciences Naval Postgraduate School Monterey, CA 93943-5000	1
5.	CDR T. J. Hoskins Computer Technology (37) Naval Postgraduate School Monterey, CA 93943-5000	1
6.	LT Richard W. Smith 1841 Dellwood Dr. Norfolk, VA 23518	2